2011.08

PROGRAMMER 程序员

海量数据

十大行业代表实践分享：Yahoo!、SAP、Teradata、百度、淘宝、麦包包、优酷、Admaster、Esri、《纽约时报》

ISSN 1672-3252

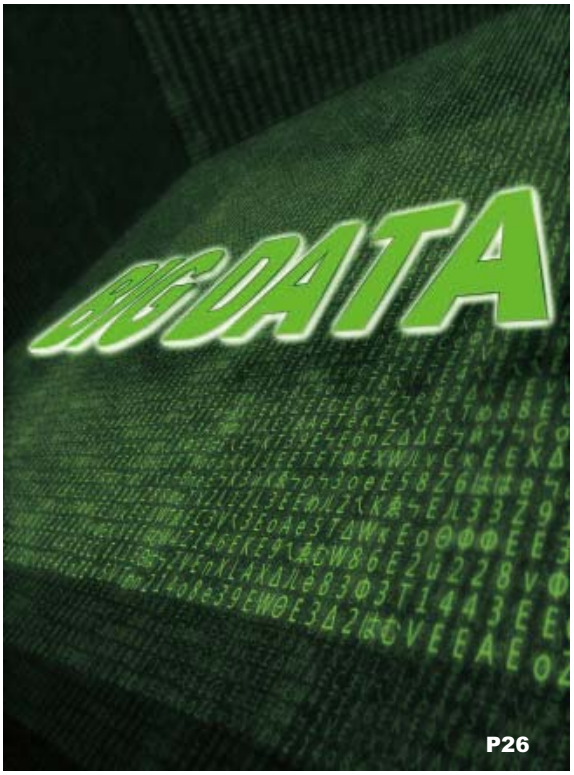
9 771672 325111 08>

邮发代号：2-665　定价：15元

www.programmer.com.cn　www.csdn.net

Contents

P10



P26

资讯

热点报道

由CSDN网站和《程序员》杂志举办的“中国软件开发者年度调查”，已历时七年。本年度调查，除了延续以往的规范化、全面深入特性外，最大的特点是紧扣2010—2011年的技术趋势和当前热点，力求准确及时地反映中国软件市场的最新变化。

封面报道：海量数据

随着互联网、移动互联网和物联网的发展，各种终端、信息收集器的数量和种类不断增加，我们每个人、世间万事万物每时每刻产生的大量数据，都在不断进入信息系统，等待存储、分析和充分利用。我们将这些浩如烟海的数据统称为海量数据，也就是国外人们更喜欢说的Big Data。本期封面报道中，将有来自支付宝、百度、Yahoo!、SAP、Teradata、新浪、淘宝、麦包包、优酷、Admaster、Esri等业内领先的软件企业和互联网企业的专家，分享他们在应对海量数据挑战方面的见解、经验和技术实战。内容涵盖运维、NoSQL、C++MapReduce框架HCE、数据魔方平台架构、电子商务推荐引擎、视频网站Big Data实践、Hadoop多维分析架构、海量空间数据库、《纽约时报》的数据可视化实践等。

Contents

2011年8月刊　总第226期

Contents

主管：中国社会科学院
主办：中国社会科学院文献信息中心
出版：《程序员》杂志社
网址：http://www.programmer.com.cn
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字0188号

总编：黄长著　Editor-in-chief：Huang Changzhu
社长/常务副总编：张悦校　President：Zhang Yuexiao
副社长：蒋 涛　Vice President：Jiang Tao
编委会：黄长著　张悦校　陈洋彬　蒋涛　曾登高　刘江
Editorial Member：Huang Changzhu　Zhang Yuexiao　Chen Yangbin
Jiang Tao　Zeng Denggao　Liu Jiang

执行主编：孟迎霞　Executive Editor-in-chief：Meng Yingxia
编辑部主任：常政　Director：Chang Zheng
责任编辑：董世晓　高松　郑柯
Editors：Dong Shixiao　Gao Song　Zheng Ke
特邀编辑：方梁　高昂　赵健平　吕娜　卢鹄翔
Contributing Editors：Fang Liang　Gao Aang　Zhao Jianping　Lv Na
Lu Dongxiang
美术设计：纪明超　Art Designer：Ji Mingchao
美术编辑：林象海　Art Editor：Lin Xianghai
流程编辑：白羽中　Coordinator：Bai Yuzhong
Tel：010-64351458
Email：editor@csdn.net

发行部　Distribution Dept.010-64351431
Email：sales@csdn.net

广告总代理：北京创新乐知信息技术有限公司
Sole Advertising Agency：Bejing CSDN Co.,Ltd
Tel：010-64376055
Email：ad@csdn.net
Marketing Dept：010-51661202（ext 149）
Email：market@csdn.net

读者服务部
Readers service Dept.
网上订购：http://dingyue.programmer.com.cn/
读者信箱：reader@csdn.net
地址：北京市朝阳区广顺北大街33号院1号楼福码大厦B座12层
Address: B-12th Floor Fairmont Tower NO.33 Guangshun North street,
Chaoyang District,Beijing
邮政编码：100102
电话：010-64351436
传真：010-64348545

法律顾问：北京中润律师事务所 王杰
Law Consultant：Beijng Hengsheng Lawyer Firm
印刷：北京盛通印刷股份有限公司
Print: Beijing Shengtong Printing Co., Ltd.
出版日期：每月1日
Publication Date：the first day per month
零售价：RMB 15.00元　新台币 390元　HK $ 35.00（港、澳）
US $ 9.00（海外）
Retail Price：RMB 15, NT$390, HK $ 35.00,US $ 9.00

并行编程

工具点评

语言

程序人生

程序春秋

名人堂

企业专栏

海量数据的意义

我曾经说过，云计算是目前最大的技术趋势，也是以互联网为基础的新一代技术的总称。宽泛地看，其中除了基础设施层面的新型硬件与数据中心、分布式计算、海量数据存储与处理等等技术之外，还包括人与人之间更多的交流方式（社会化网络），终端设备的多样化（移动），无所不在的数据采集方式（物联网），和新一代自然用户界面、用户体验。

其中，海量数据将发挥核心作用。

互联网企业与传统行业（包括软硬件和电信厂商）相比最大的优势，就是几乎消除了任何中间层，所有最终用户使用产品的行为数据可以轻易地保存在服务器上；通过对这些海量数据的挖掘、分析和图形化呈现，能非常清晰地揭示用户的行为模式，加深对用户需求的理解，提取用户的集体智慧，从而为产品研发人员决策提供依据，同时不断提高系统的智能化水平，提升产品用户体验。而社会化网络更进一步，用户都以登录甚至实名为主，可以为每个人提供个性化的服务。

早在2005年，Web 2.0的主要倡导者Tim O'Reilly在经典文章“什么是Web 2.0”中就强调过“数据是新的Intel Inside”，也就是说，就像PC时代Intel芯片是核心一样，数据是新一代计算的核心。（细读此文，我们会发现云计算与Web 2.0之间的紧密关系。）

2007年Google承认，在很多情况下海量的数据比好的搜索算法还要重要。而它的竞争对手近来也在抱怨彼此的差距主要在用户搜索数据的积累上。Google的很多产品，比如翻译和语音输入，同样得益于海量语料库的支持。2009年Google的研究总监Peter Norvig等人发表了《The Unreasonable Effectiveness of Data》一文，得出一个结论：简单的模型加上海量的数据比精巧的模型加上较少的数据更有效。也许正是因为如此，Google首席经济学家Hal Varian才会坚称数据科学家将是未来十年最具吸引力的职位，他认为管理者甚至中小学生，都应该具备对数据进行处理、从中提取洞察、理解和表达的能力。

Facebook更是众所周知的数据驱动无所不在的公司。2006年，为了找出Facebook在某些学校不受欢迎的原因，公司从华尔街聘请了数据科学家Jeff Hammerbacher（现为Cloudera的首席科学家），他和其他几位同事组成了最早的数据团队，在不知不觉中自行研发了一个商业智能系统。这成为Facebook日后产品成功的重要基础，不仅很好地支撑着工程团队，而且在产品设计决策中也起到了关键作用——他们的设计师也能写代码，用真实内容和页面来做产品原型，然后上线测试，快速迭代，2009年时设计团队每周提交40次。

可以想象，随着云计算的发展，世间每个人每件东西每天每秒所产生的新数据（位置、状态，所见、所闻、所思、所言……）都有能够被更完整和丰富地数字化，并全部联入互联网。近年来互联网领域的创新几乎都可以纳入这一模式：Google将每次搜索和去向存储下来，Facebook将人们之间的各种交互数字化，Amazon、淘宝等电子商务网站将商品流通的信息、人们的购买喜好数字化，Twitter等微博将人们稍纵即逝的想法和谈话记录下来，Foursquare等LBS应用将人们去过那里的信息记录下来，Instagram、Color等移动应用捕捉瞬间的图片和音频、视频……

这其中蕴含的巨大潜力将是空前的，影响极为深远。**互联网企业甚至更多行业企业之间的竞争将围绕谁拥有更多详细的用户数据并能善加运用展开**。我们很有可能打造一个智能系统，实现计算机科学长久以来的人工智能梦想。

邮件：liujiang@csdn.net
新浪微博：@刘江CE
欢迎大家交流反馈，欢迎投稿、批评、建议和挑错

淘宝技术嘉年华

2011年7月9日～10日，淘宝技术嘉年华（Taobao open developer conference，简称TDC）在杭州举行。TDC大会每年举办一届，本届TDC大会以“共建、共享”为主题，七十余场的主题演讲，十几场技术研讨沙龙，汇聚了以D2、iConference、iDataForum、TCon、ACT设计论坛为代表的专业品牌会议，更有淘宝技术专场、开放式技术沙龙等引人入胜的活动。主办方除了发动淘宝的技术专家到会交流，更是力邀来自Yahoo!奇摩、无名小站、腾讯、百度、豆瓣、口碑网、支付宝、微软等著名公司的技术高手，一同分享技术话题。

会中的很多技术演讲受到现场听众的格外关注，如淘宝数据平台开发团队技术专家林昊就Hbase的特性，在淘宝中的应用及发展趋势做了探讨；盛大工程师魏子钧就HTML5在网页游戏中的应用做了诸多生动的演示；争渡读屏开发团队的盲人开发工程师杨永全的演讲，让很多现场观众第一次了解到，网页内容无障碍体验与设计是多么的重要；淘宝网高级技术专家华黎总述了用Java构建有特色的业务系统，为观众全面了解淘宝的技术特点提供了一流的视角等等。现场的演讲精彩纷呈，主办方也将演讲资料陆续在网上发布，将开放办会的心态贯彻到底。

此次大会之后，淘宝还计划推出技术嘉年华系列活动，如月度技术沙龙等，让工程师们能够有形式多样的互动交流，在国内推动形成一个良好的技术交流氛围。沙龙的精彩分享也会选作每年技术嘉年华的预选话题。

团购领域前赴后继

7月，Groupon提交了修改后的IPO申请文件，披露一季度财务数据，其营收达6.4亿美元，净亏逾1亿美元，但Groupon更多强调了公司海外扩张面临的风险。

同时，手机业务自顾不暇的诺基亚也打算推出名为Perks的团购服务，以吸引大量的手机用户。连标杆企业的亏损，也阻挡不了后来者的步伐。

这边厢，国内的团购网站开始准备过冬。有寅食卯粮的，糯米网披露了第一季度财报：收入90万美元，而其运营成本为460万美元。有信心满满的，美团网宣布已经完成B轮5000万美元融资，CEO王兴甚至晒起了银行账户。有逆势而入的，原本的IT产品发行商数字星空也高调宣布团购业务上线，自此杀入团购领域。互联网泡沫的警告言犹在耳，百团大战硝烟未散，大家都挤在团购里，难道真是此地人傻钱多？须知大量扛不住寒潮的企业在自己倒掉的同时，也必将削减消费者对此行业的信心，盲目入市的结果终是损人害己。

IBM–WISCO创新中心与钢信软件落户武汉

7月12日，IBM-WISCO创新中心暨武汉钢信软件有限公司（简称钢信软件，WIST）揭牌仪式在武钢隆重举行，据悉，成立后的钢信软件将充分发挥双方的优势与整合能力，传承双方的行业经验与专业知识，利用各自品牌优势和技术实力，共同专注于大型企业的信息化咨询、IT服务外包业务，建立一套适合中国钢铁及制造业的信息化解决方案。

微软发力商务零售解决方案

6月17日，微软针对上半年新推出的商务解决方案Microsoft Dynamics AX for Retail，在北京举行了媒体见面会。微软商务解决方案事业部零售行业全球产品总监Michael Griffiths向本刊记者表示，该方案特色之一就是提供集成系统，包括前端的POS管理系统、中间的总部营销系统以及后端的ERP系统，实现定期的数据统一传输及实时的数据传输。Michael Griffith说：“微软希望能够充分利用我们对行业的理解以及成熟广泛的技术，为行业用户提供集成的体验。这样可以帮助客户更好地利用现有的平台投资，简化管理从而能够提高企业竞争力。”

“最后主流的视频网站将会是五家，搜狐视频会是其中一家”

——7月13日，搜狐公司董事局主席兼首席执行官张朝阳在“闭关”近半年后，重新面对媒体，并首次明确表示，搜狐视频未来单独分拆上市的可能性。

“我四十岁前已经干了不少事：卓越卖了、金山上市了、天使投资也不错，但我迷茫了：18岁的理想一直没有实现，觉得心里不踏实，计划悄悄干他一年半载的，如果输了，这辈子就彻底踏实了。这样创办了小米，十五个月过去了，今天终于鼓起勇气出来了，请大家批评吧。”

——在接受新浪微博微访谈中，雷军表示，小米手机将完全是自主独立研发再找工厂代工。

“这种应用、智能手机、平板电脑和云的泡沫仍在继续膨胀，那些现在抱怨其估值的家伙将带领这些公司以过高的估值上市，而这种趋势很可能不会停止。”

——美国前对冲基金经理Cody Willard在MarketWatch上撰文称，Facebook被估值过高。

“所有的谷歌员工都希望创建全世界的人每天都会使用两次的服务——就像牙刷一样。”

——谷歌CEO Larry Page在出席财报电话会议时通过Google+同步发布的会议摘要中，如是说。

“最好的情况，G+可以挑战Facebook霸权，或者逼迫Facebook开放。最差的情况G+也会是社交类的一个生存者(Robert Scoble说最糟G+就会是一个极客社区)。”

——7月13日，创新工场CEO李开复撰文对谷歌新推出的社交服务Google+进行分析。

“这个市场不缺商品、不缺卖家，移动电子商务最多在三年内就可以赶超传统电子商务十年的积累。”

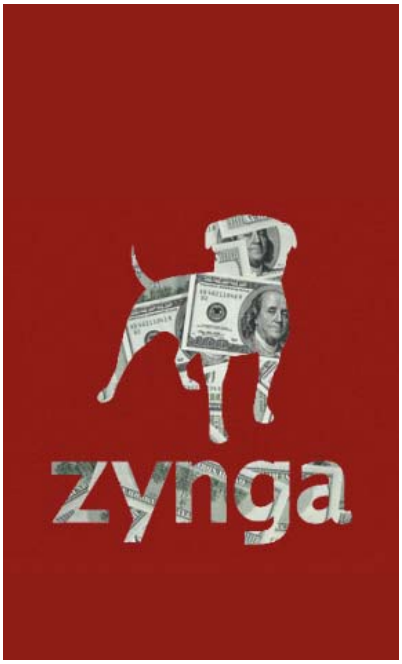
——淘宝网副总裁邱昌恒他对移动电子商务做如是点评。

“诺基亚是被两家台湾公司打下来的，一家是联发科，另一家是HTC。”

——诺基亚，2008年市占率高达全球4成的手机霸主，在短短4年之间，市值暴跌87%，债券评级几近垃圾等级，一个有146年历史的领袖企业如何濒临崩坏？台大副校长汤明哲如此观察。

CSDN十大新闻

2011年6月26日~7月26日



01 Google封杀1100万CO.CC域名

作者 / Aria

过去几个月，Google检查了大量子域名提供商，它们常因恶意滥用而成为攻击目标。其中，Google全面封杀了提供恶意域名的CO.CC。此域名由一家韩国IDC公司提供，申请量巨大，是免费顶级域名。据Anti-Phishing Working Group统计，2010年下半年，CC域名发起4963宗钓鱼攻击。

02 成就亿万富翁的10条规则

作者 / Robert Frank

域名注册、网站空间租赁公司GoDaddy的创始人Bob Parsons因潜在的收购，将成为亿万富翁。WSJ为此撰文，揭秘成就辉煌的10条规则。其中一些相当老生常谈，譬如永不放弃；永远向前迈进；不要期望生活公平；不要让自己太严肃等等，当然，知易行难。

03 Mayer：要和最聪明的人一起工作

作者 / Bianca Bosker

本文介绍了Google高级副总裁Marissa Mayer的求学求职经历。谈到选择工作，Mayer总结说要与最聪明的人合作、做从未预想的事、找到舒适的环境并为信任你的人工作。此外Mayer认为偏见导致科技领域鲜有女性，但随着技术影响力的扩大，这一状况将得以改善。

04 Kinect：微软“无心插柳柳成荫”

作者 / 萧萧

Kinect成为有史以来销售速度最快的消费类电子产品，还是最热门的设计平台。许多人用Kinect制造创意翻新的新装置。对此，微软发布了Kinect软件开发包，方便给开发者使用。此种做法类似于Google对Android，两家公司都不知道其产品会成为发明和创新的平台。

05 微软可能放弃Windows品牌

作者 / 木秀林

微软全球合作伙伴会议上，微软高级副总裁Andy Lees称：“可以在任何设备上拥有完整的PC计算能力，（微软）不会为PC、手机和平板电脑分别建立一个生态系统”。这意味着四年后，我们将见证同时运行在PC、平板电脑、手机和Xbox上的超级操作系统。

06 Google+成就卓越需做七件事

作者 / Jon Mulholland

Google+推出后迅速得到业内人士的赞扬，然而要想成就卓越，Google+还有七件事要做：发布iPad应用；建立各类团体主页；自动导入其他社交媒体源内容；允许创建或提交内容到Sparks中；提供浏览器书签；将+1迁移到Stream；将Circles应用到Google所有产品中。

07 Zynga的200亿估值：泡沫几何

作者 / 杨樱 骆轶航

7月2日，Zynga正式申请IPO，成为今年最有实力的IPO公司，但没有明显的核心技术、对Facebook和AWS的致命依赖，营收带狭窄，利润率起伏不定，硬件投入对成本的提升等不利因素，也让人怀疑，Zynga会确立数据处理公司的江湖新地位，还是新一轮趋势性公司失意泡沫的开始?

08 Twitter所带网络流量超过想象4倍

作者 / Mark Suster

GRP Partners合伙人、BothSidesoftheTable创始人Mark Suster在一篇综述里，解释了应用Jonathan Strauss设计的awe.sm对网络流量统计的分析方法，并强调了Twitter在网络推荐和推广中的影响力是重要的“最后一英里的归属性”。他的分析的结果证明了事实永远没有数据所显示的那么简单。

09 微软CEO鲍尔默坦承WP7未获成功

作者 / 晁晖

微软首席执行官Steve Ballmer在“2011年全球合作伙伴大会”上，承认公司许多产品在过去一年获得了成功，但其中不包括WP7。谈到WP7的市场份额变化时，鲍尔默说，“一年来，我们的市场份额依然非常低。”Nielsen调查数据显示，WP7在美国的市场份额仅为1%。

10 阿里云多款产品终极内测

作者 / 付江

继7月初阿里巴巴旗下的阿里云计算新官网上线，自主研发的阿里云手机操作系统、阿里云浏览器在集团内部终极内测以及包括浏览器、输入法、地图、翻译等基础应用的截图相继曝光后，阿里巴巴在无线领域的云布局已经逐渐浮出水面。

2011财富500强十大IT企业

营业收入（百万美元） 利润（百万美元） 国家

前100强中共有16家科技公司上榜， 其中，三星排名攀升10位，超越惠普一举成为全球最大的IT公司（按营收排名）。而苹果营收去年增长了79%，增幅远远超过其他上榜IT公司。在中国科技公司中，富士康母公司鸿海科技排名居首（第60位，年营收952亿美元）

01 三星

133,780.5 13,668.7 韩国

主营业务：消费型电子、DRAM 与 NAND Flash，微控制器和微处理器、无线通信芯片与晶圆代工。

02 惠普

126,033 8,761 美国

主营业务：打印机、数字影像、软件、计算机和咨询服务。

03 日立

108,766.4 2,788.9 日本

主营业务：家用电器、电脑产品、半导体、产业机械。

04 西门子

102,657.2 5,268 德国

涉及自动化与控制、动力、交通、信息通讯等。

05 松下

101,491 864.2 日本

除家电外，还涉及数码产品，并可扩展到电子零件、电工零件、半导体等。

06 IBM

99,870 14,833 美国

大型／小型机和便携机，转型后主要注意力放在服务管理以及咨询业务。

07 鸿海

95,190.5 2,450.4 中国台湾

研发生产精密电器元件、机壳、准系统、系统组装、光通讯元件、液晶显示器等3C产品上下游产品以及服务。

08 索尼

83,844.7 -3,030.8 日本

主营电子产品，特别是电子娱乐产品。

09 东芝

74,705.5 1,609.4 日本

半导体以及综合电机制造。

10 苹果

65,225 14,013 美国

主要产品包括Macintosh电脑、iPod、iTunes商店、iPhone和iPad等。

苹果杀软

Intego放出了专为苹果iOS设备定制的杀毒软件VirusBarrier。

这个软件可以让iPhone，iPad和iPod Touch的用户查杀“Windows和MacOS所有已知的病毒，包括病毒、蠕虫、特洛伊木马和伪杀毒软件等威胁。”

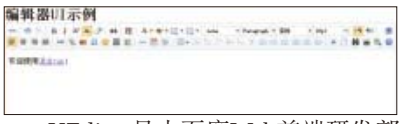
这个软件也可以检查压缩文件，带毒的PDF文件，电子邮件附件，下载文件等，还能修复受损的系统文件。当然，也会连接到Intego的网站更新病毒库。

虽然，PC上的一些功能，比如自动运行，定时查杀，未知病毒查杀等功能尚未实现，但是这毫无疑问是个好的开始。

领域专用语言开源框架Xtext 2.0 发布

Xtext 2.0近日作为Eclipse Indigo的一部分发布了，新版提供了一个新的重构框架，一个新的表达式语言和一个新的静态键入模板语言。除修正了上百个已知问题和增强性能外，Xtext2.0还增加了以下新功能：重构框架及支持富悬浮（Rich Hover）信息；可以嵌入在任何一种DSL中的新表达式语言，以及允许用户在自己的语言中写计算逻辑；一个静态键入模板语言Xtend，该语言简化了代码生成器的开发和维护，它已经集成在Eclipse工具中。

百度UEditor



UEditor是由百度Web前端研发部开发的所见即所得富文本Web编辑器。

这款所见即所得的编辑器由百度Web前端研发部开发，具有轻量，可定制，注重用户体验等特点。

能降低网站的开发成本，尤其在代码部署和定制化开发方面提供解决方案。

UEditor基于BSD开源协议，除了具有代码精简、加载迅速的轻量级特质外，还采用了分层理念，使开发者可以根据实际应用和需求自由定制。

Ueditor编辑器划分为了三层架构。其中，核心层为开发者提供了诸如range、selection、domUtils类的底层API接口，中间的命令插件层不仅提供了大量的基础command，还允许开发者基于核心层进行command命令的开发，而面向用户端的界面层则可 以提供自由定制的用户交互界面。

Ueditor开源编辑器这种拥有可配性的模式，令开发者能够根据自身需要接入任何一层进行开发。

百度透露，到目前为止，已经针对Ueditor进行了上千个自动化测试用例和手动化测试用例，并在此基础上对产品进行完善和升级。

微软发布最终版Windows服务器手机连接器

微软在今年初发布了这个软件的候选发布版。最终版本能够与安装微软Windows家庭服务器2011操作系统的家庭服务器一起工作。它要求下载这个服务器操作系统以及一个手机应用程序。微软的在线Zune商城可以找到这个软件。一旦安装了这两个软件，用户就可以设置自己的WP7手机访问和查看远程存储在家庭服务器中的图片、音频或者视频等任何媒体文件。用户还可以使用手机管理Windows家庭服务器中的事情，如设置文件备份时间以及管理能够访问这个服务器的用户数量等等。

新浪开源编辑器

新浪开源编辑器目前代码托管在Google Code上，目前SinaEditor暂不支持IE浏览器，据开发者称，将会引入另一个开源项目ierange，并进行二次开发。此功能将会在稍后的版本中完善此功能。

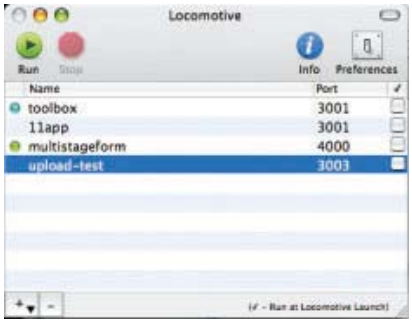
PuTTY 0.61发布

经历四年开发之后，PuTTY的版本号从0.60增加到了0.61。PuTTY是开源终端模拟器，可用作Telnet／SSH／rlogin／纯TCP以及串行阜的客户端。0.61主要是修正bug，改进性能，增加新特性，支持GSSAPI SSH2认证，更快的SSH密钥交换，兼容Windows 7，支持Win7的Jump List，等等。

崭新内容管理系统Locomotive

Locomotive是基于Rails3版本、数据库MongoDB开发的内容管理系统引擎；作为一款基于云的开源搜索引擎，它能够一次安装简单搞定不同域名网站内容。

Locomotive采用了大量最新的Ruby软件包，如 CarrierWave 、 Devise 和 Liquid 等。



阿里巴巴开源平台新增项目Druid

Druid是一个JDBC组件，它包括三部分： DruidDriver 代理Driver，能够提供基于Filter－Chain模式的插件体系；DruidDataSource 高效可管理的数据库连接池；SQLParser应用Druid可以监控数据库访问性能，Druid内置提供了一个功能强大的StatFilter插件，能够详细统计SQL的执行性能，这对于线上分析数据库访问性能有帮助；替换DBCP和C3P0。Druid提供了一个高效、功能强大、可扩展性好的数据库连接池；实现数据库密码加密。直接把数据库密码写在配置文件中，这是不好的行为，容易导致安全问题。DruidDruiver和DruidDataSource都支持PasswordCallback。

Druid提供了不同的LogFilter，能够支持Common-Logging、Log4j和JdkLog，可以按需要选择相应的LogFilter，监控应用的数据库访问情况；扩展JDBC，如果你要对JDBC层有编程的需求，可以通过Druid提供的Filter-Chain机制，很方便编写JDBC层的扩展插件。

目前Druid是一个开源项目，基于Apache License 2.0。

微软发布Small Basic 1.0正式版

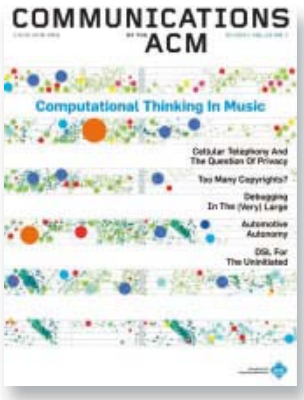
经过两年的测试，微软终于发布了Small Basic 1.0正式版。微软自己也承认，其中仍然有很多功能还有待完善，不过Small Basic的核心功能已经基本完善，能够满足新手的使用了。

Small Basic 1.0在0.95版本的基础上进行了小幅升级，在过去几个月里开发团队主要对其在线体验进行了完善，并进行了大量的翻译工作。。

Small Basic 1.0更新内容包括，一共支持20种语言；升级了Small Basic启动程序和版本信息。另外，Small Basic 1.0还新增了不少扩展，用户可以到TeachingKidsProgramming网站查看。这里还提供了不少课件，帮助10岁以上儿童学习编程。

RSS阅读器RSSOwl 2.1 发布

RSSOwl是一个完全用JavaSWT开发的RSS/RDF/Atom新闻阅读器。按类别读取新闻，分类存储收藏信息，导出为PDF/RTF/HTML/OPML，从OPML格式导入，支持全文本搜索，使用Internet浏览器。该版本新增一些很酷的特性，例如与Google阅读器的文章同步、新的头条和列表布局、重新设计的报纸视图、新闻打包。另外，该版本在性能上和可靠性上也有提升。



ommunications o
2011.0

DSL

软件项目失败的主要原因之一，是因为用户和开发者之间缺乏沟通。DSL可以借助共享的词汇表促进更好的协作，从而填平用户和开发者之间的语义鸿沟。在建立某个领域的模型时，开发者使用和用户一样的术语，这样DSL的抽象就能够符合问题领域的语法和语义。在项目的整个生命周期中，用户也能随时参与到对业务规则（Business Rule）的验证中来。

设计DSL的目的是让程序中的业务规则更加清晰明了。DSL的优点包括：可以更简便地与用户协作；更好地表达领域规则；基于DSL的API涉及的代码更少；基于DSL的开发具有可扩展性。虽然有这些优点，但是设计不合理的DSL也会带来很多麻烦，其缺点包括：入手困难；前期投入大；多种语言并用带来的问题。

根据实现的途径，DSL可以分成两类：其中一类在已有的编程语言基础上实现，称为内部DSL或者嵌入DSL；第二类DSL则是独立的语言，不借助现有的宿主语言的基础，而拥有自己本身的语法、语义以及语言基础结构，称为外部DSL或者独立DSL。

本文主要对嵌入DSL进行了讨论。

DSL的基本结构是语义模型（领域模型）之上的语言抽象。使用宿主语言的习语设计领域模型，该领域模型被基本抽象引用。DSL建立在基本抽象之上，但基本抽象的实施部署独立于DSL。这样，可以实现在一个领域模型上部署多个DSL。在DSL代码和语义模型之间，增加一个自定义的解释器，这样可以实现模型和宿主语言的分离。

开发嵌入DSL时，首先要选择一种合适的宿主语言。宿主语言越有表现力，为DSL而建立的抽象和该语言原生抽象之间的语义鸿沟就越小；然后要建立领域模型。领域模型要包括业务的核心抽象。在该领域模型上建立的DSL要能够简洁明了地表达业务规则，并且方便用户核查。

嵌入DSL鼓励在更高层次的抽象上进行编程，让使用者可以专注在业务功能的构建和该领域语法语义的使用上，因此能够提高效率。

寻找Jim Gray

2007年1月28日，著名的计算机科学家Jim Gray和他驾驶的帆船在海上失踪。从而引发了一次史无前例的平民搜救（SAR）行动，动用了卫星，私人飞机，自动图像分析，海流仿真，以及众包人类计算（human computing）。救援团队的成员来自各行各业，而且素未谋面。行动所需的资金、技术、技能也来自不同渠道。

虽然这次救援以失败告终，但通过回顾这次行动，总结其中使用到的技术，并思考如何改进它们，可以作为借鉴。也希望能对更重大的危机事件有所启示，比如自然灾害和军事冲突中平民参与的救援行动。

在对Gray的搜救中，沟通和协调是很重要的两个部分。救援组织结构的演变是很有趣的一个方面，通过电子邮件沟通，自然地分化出了领导者与各司其职的技术分支小组。另外，在这次事件中，涉及了多种沟通方式的应用：电子邮件、博客、播客、维基、邮件列表等等，这些技术的混合使用给沟通和协调带来了挑战。信息管理技术可以解决大部分的问题，但是仍然需要更好的软件来促进群组中的沟通和协调。

搜救中，短期内成功获得了87G的卫星图像，但实际上，需要在更短的时间内获取更多的数据。而政策因素以及隐私问题阻碍了大量远程图像的获取。

获取图像数据后，需要对原始图像数据进行处理，选定合乎要求的搜索区域，再用飞机进行近距离搜索。这是整个救援行动中最具技术性的一个步骤。其中涉及了几个主题：

综合设计。行动中，软件开发和部署过程是由一组独立工作的专家共同完成的。这个过程中的冗余保证了可用性和可靠性。另外在这个连接松散的软件开发过程中，出现了各种接口，来整合这些独立的部件。这种综合技术设计能成为可能，得益于近十年来Web的发展，特别是数据公开和全球传播之间的相互作用。

网络化搜索。以网络为中心的SAR方法在规模扩展和演化上具有一定的优势。这提出了一种混合方法，其中搜索过程的相关部件是分离的，但是能够更加耐心地加以架构、演化和整合。在军事和医疗中，各部件和专业知识的联网已经变得越来越普遍。将这种想法应用在SAR中，定能获益匪浅。

自动化图像分析。在搜索中，计算机视觉专家发现现有的图像识别软件无法处理获得的卫星图像数据，从公共海域的卫星图像上辨别出船只。SAR行动中对远程图像的应用是计算机视觉专家需要解决的一个问题。

工作负载管理

很多评估都表明，功率成本会成为数据中心运营总成本中，最大的单项支出。相反地，还有一些研究表明，数据

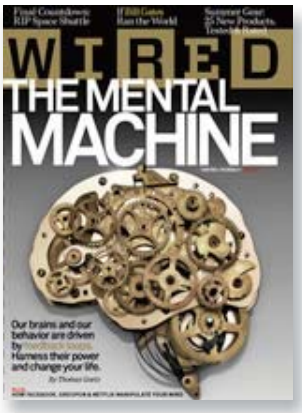
中心里的服务器未充分利用的情况也很严重。这是一台服务器为预备不常出现但无法避免的工作负载高峰的结果。具备功率监控的动态应用配置可以解决服务器的不充分利用问题，以及数据中心上涨的功率成本，通过迁移应用来更好地利用服务器并且将空余的服务器切换到低能耗状态。

动态应用配置已经不是新概念了，但目前的两个趋势，即现代服务器中的虚拟化和能量管理技术让它在数据中心中得到了广泛的应用。其中，虚拟化使其成为可能，而功率最小化则是最主要的驱动因素。

由于动态整合，数据中心需要不断重构，这让性能和功率建模成为一项具有挑战性的任务。几个基本的挑战包括：虚拟化天花板；同址（co-location）天花板；不同系统之间的工作负载归一化；多个同址应用的功率建模。

之前，工作负载管理的目的是在降低数据中心运行费用的同时，最大化数据中心获得的收益。虚拟化数据中心里，工作负载管理主要着重在应用或虚拟机在实体服务器上的静态一次性放置。这种配置主要是根据可行性进行的，而不是效率。印度IBM研究院开发的工作负载管理工具Emerald，着眼于数据中心整体成本的最小化，同时满足性能要求。其关键部件包括：读取装置，从基础设施中读取监测数据；平滑器，过滤掉原始数据中的噪音。归一化装置，目标和源服务器在CPU处理能力上有很大不同，需要对服务器配置进行准确的归一化； 规模估计和工作负载衡量；限定条件；配置引擎；投资回报率评估器。

Emerald的设计保证了每个部件高度定制化，而且附加逻辑可以很简单地合并，而不需要改变核心接口。所有基于虚拟化的工作负载整合的主要目标是尽可能地建立最优的整合路线图，遵循一定的已知限定条件。包括技术限制和商业限制。Emerald使用的两阶段算法pMapper能够在最小化迁移成本的同时达到最低的存储碎片水平。其投资回报率评估器可以评估整合前后的动态和静态成本。



Wired
2011.0

互联网公司，如何引诱用户交出信息和金钱

你不是傻瓜，可还是免不了会被愚弄。优秀的营销者知道如何利用人性的弱点。借鉴心理学的观点，行为经济学解释了为什么0.99元的定价更能刺激我们的购买欲，为什么我们会购买根本用不到的健身会员卡。从亚马逊到Zynga的互联网巨头，使用了同样的诡计，吸引我们访问网站、玩游戏，购买商品。

亚马逊：减少小摩擦可以改变一个人的决定。这就是默认选项的力量：我们倾向于选择阻力最小的那条路。亚马逊默认地记忆我们的信用卡和地址信息，减少了用户重复填写信息的麻烦。在运费上，亚马逊也做出了两个聪明的举动：第一是优惠送货，25美元以上免运费，这让人们消费更多书和CD来减免邮费。还有亚马逊Prime，79美元一年的免运费服务，为了自己的投资物有所值，用户会更多地在亚马逊上购买产品。

Netflix：这家公司10亿美元的生意建立在一个简单的原则上：人们痛恨滞纳金。免除滞纳金的政策消除了用户的顾虑。此外， Netflix还建立了一个智能系统，向用户推荐他们可能会喜欢的电影。

Groupon：Groupon这种社交购买网站的革新之处并不在于能够提供很大的折扣。而是为其目标人群免除了使用优惠券消费的尴尬。实际上，使用优惠券的耻辱感是确实存在而且很普遍的。而从众心态是很强大的一种动力，可以改变人们的这种感觉。Groupon的限时政策是另外一个秘密武器。人们不喜欢后悔的感觉，但如果他们不马上做决定，就有后悔的可能。

Zynga：一个创造过程越复杂、越困难、越花时间，我们就会越加珍惜自己创造的成果，这就是Zynga的“开心农场”和其他游戏的魅力所在。而社交元素是另一种强制力量。互惠主义也是一种很强大的动力。人们给你帮助，你会想办法回报。在“开心农场”中，很多活动都和互惠主义有关，这一点也让你花费越来越多的时间在游戏中。

Facebook：Facebook的诀窍都是围绕信息墙（The Wall）开展的：你会忍不住要回复“墙”上的信息。我们也希望能通过“墙”展示自己，因此产生一种动力，经常地能够反映出自己检查和维护它。但Facebook最吸引人的地方也许是

它能够以相对便宜的方式提高一个人的地位。

苹果公司：如果你是苹果的用户，你可能会注意到在购物时往往会有几个小时，甚至几天的延迟。这种延迟减少了经济学家口中的“支付的痛苦”。支付和消费的不同步减少了支付的痛苦。但是在定价上苹果公司犯了错误：它的应用程序卖得太便宜。有一种经济现象叫做“锚定效应”，消费者的心理价位一旦被第一次的价格所锚定，就很难再改变了。

面对这些互联网世界的愚弄，我们能怪的却只有自己。以电子邮件为例，近几年，大家仿佛都认定每个人都应该随时电子邮件在线。对大部分人来说，这不是件好事情。心理学家发现，随机的强化比规律的强化更能够改变人的行为。我们不时地会收到非常重要的邮件，因此我们常常等候新邮件的到来，从而强迫自己阅读新邮件，最后却发现并不是什么重要的邮件。我们对自己的愚弄，还和最古老的一项技术：日历有关。日历上，我们没有安排日程的时间都显示空白，因此强迫我们不断地用不必要的事情去填满它。

inect黑客，改变机器人学的未来

25年来，机器人领域最基本的问题是：要做到行动自如，机器人必须建立一个反应周边环境的地图，并感知自身在其中的位置。也就是同步定位与地图创建技术（SLAM）。机器人专家建立了各种工具来完成这项任务，但是传统的传感器不是贵且笨重，就是便宜却不精确。2010年12月4日，一款电视游戏为这个问题提供了解决方案。这一天，微软公司发布了Kindle，一款标价150美元的Xbox附加设备，允许玩家通过移动自己的身体指挥游戏中的行动。全世界都在关注它独特的体感交互界面，但是机器人专家眼中看到的是一款价格合理的轻型摄像头，可以实时捕捉3D图像。这款产品发布后几周之内，Youtube上就充满了使用Kinect支持的机器人视频。

机器人爱好者不是唯一发现Kinect的无限潜力的人。研究者、视觉艺术家、色情文学作家都开始创建Kinect项目，并且在网上发布自己的成果。这些项目都没有得到微软的支持。事实上，几个月之前，必须安装由一组黑客自制的驱动，才能将Kinect用于xbox之外的用途。不过很快，2011年6月份，微软公司发布了一款软件开发套件，允许学术研究者和爱好者利用Kinect的摄像头和麦克风创建Windows应用程序。同时准许开发者接触到Kinect的强大算法，这种算法能够让机器识别出人的身体和跟踪其运动。

大的制造商很早之前就意识到允许用户修改其产品的价值。很多成功的技术公司鼓励独立的开发者在他们的平台上进行开发——比如说Windows，Facebook和iPhone应用程序商店。历经多年，修改者们引进了很多创新，最后成长成了完整的产品类别。但也有一些公司，对这种hack行为采取置之不理或者法律诉讼的处理方式。但是迄今为止，没有一家公司在像Kinect这样的热门产品上，对黑客采取如此欢迎的态度。

另一方面，微软也没有坐等黑客开发Kinect的巨大潜力。这家公司在kinect的基础上开展了多项研究，包括可以触觉感知的全息图像、可以为不同观看者展示不同图像的LCD屏幕。同时还准备发布Kinect的第一个重大软件升级，可以对人的面部表情进行捕捉。

随着越来越多的技术被商品化，还有网络让世界各地的人们越来越方便的协同工作，黑客社区也会越来越强大。数量众多的爱好者将会更快地拿出更有趣的想法，远远胜过专业的工程师。

反馈回路方法的能量

反馈回路（feedback loop）是一种改变人们行为的有效方法。基本的方法很简单，即实时向他们提供行为的信息，给他们改变行为的机会，促使他们做出更优的举动。行动——信息——反应。

但是反馈回路并不是看上去那么简单。实际上，这是一个很强大的工具，可以帮助人们改掉坏的行为模式。同样可以将进展本身作为一种奖励，用来鼓励好的行为。拜新科技爆炸所赐，如今我们将其付诸于实践，应用于生活的方方面面。

一个反馈回路包括四个不同的阶段。第一个是数据：行为必须被测量、捕捉、存储。这是搜集证据的阶段。第二个，信息必须被传播到个体，不是以原始数据的方式，而是能够引起情感共鸣的内容。这是寻找相关性的阶段。第三个阶段是推论阶段。这些信息必须引导向一种或多种途径。第四个阶段是反应阶段。必须有一个明确的时刻，个体可以重新校准行为，做出选择和行动。然后行动会被测量，再进行一次反馈回路。

反馈回路的基本框架很早就被提出，但它一直没有在日常生活中得到应用。直到最近几年，传感器的价格急遽下降，引发了一场反馈回路的革命。反馈回路现在被应用在各个领域，Rypple的在线平台可以帮助工作者做出或接收反馈。Zeo的头戴受话器可以测量与睡眠质量有关的脑电波，并且通过床头显示器展示用户的睡眠状况。Belkin生产了一种简单的插接设备，用来测量任何设备的能耗。GreenRoad的车载显示器利用GPS和加速计帮助用户注意到危险或者造成燃料低效的驾驶习惯，并且改正。GreenGoose使用无线传感器和简单的游戏来鼓励良好的行为。

与传统的方法相比，反馈回路改变行为的效果通常能提高10%。一款提醒病人用药的设备GlowCaps，甚至能将效果提高40%。

（感谢译者卞斌支持）

年度编程语言锁定 Objective-C

虽然2011年才刚过半，但随着iPhone和iPad平台市场领导地位的确立和Apple各系统不断推陈出新，从市场占有的趋势来看，Objective–C成为年度编程语言无疑。

TIOBE编程语言排行榜每年初都将统计评出上年度的最佳编程语言大奖，获奖的编程语言被认为在该年度市场份额增长及用户认可度方面获得了最为突出的成绩。

历年获奖的编程语言包括2004年的PHP、2005年的Java、2006年的Ruby、2008年的C和2009年的Go语言。成为系统脚本事实标准的Python语言，因其简单易用并能成功运用在Web开发等各种不同类型的应用中，而于2010年和2007年两次获得年度编程语言奖项。

2011年已过去一半，从目前市场占有趋势来看，Objective-C已经脱颖而出，相对2010年的市场占有率增长了2.68%，毋庸置疑，Objective-C将成为年度编程语言奖项的得主。其实在2010年，在iOS平台应用快速增长的推动下，Objective-C已成为上年度使用率上升最快（增速超过100%）的语言，但从绝对数量来看，Python以拥有最大的市场份额增长率而胜出。

Objective-C设计的初衷是在C语言基础上添加面向对象拓展，目前Objective-C已发展成为苹果各系统平台上应用开发的事实标准。Objective-C包含一个用C编写的小巧的运行库，提供给开发者各种实用的操作。Objective-C的编译可以在现存C编译器基础上实现，而不需专门开发一个全新的编译工具。这让Objective-C能重用大量遗留的C代码、类库等各种资源。

目前Objective-C代码的编译在GCC和LLVM/Clang中已得到完美支持，并且GCC的最新4.6版本还支持多项Objective-C 2.0语言的新特性，比如说快速枚举、dot语法、可选协议方法、类拓展以及新的Objective-C运行时API等。

Objective-C的独特之处在于，它将垃圾回收等一些真正高层次的语言特性与C语言低层级的函数功能相结合，通过高层次的语言功能提高程序员的生产力。但即便如此，不少开发者仍旧对语言的发展给予厚望。

Mac OS应用Illuminate的作者Andy Finnell在他的Blog中写道，希望在下一个Objective-C 3.0版本发布时，增加闭包的支持，并增加匿名函数以方便在函数中与前端JavaScript交互，并提供在匿名函数中访问本地变量的功能。

在公司或个人准备启动一个新的软件开发项目时，TIOBE编程语言流行度排行常被用作检测编程语言或技术先进性的参照尺度，来辅助技术选型的决策。苹果iPhone和iPad平台市场领导地位的确立和Apple各系统不断推陈出新，极大的推进了Objective-C语言的普及和流行，使其成为TIOBE本年度最佳编程语言。

随着苹果公司Mac OS X Lion和iOS 5.0发布日期的临近，具备更强劲性能的硬件和更丰富易用的平台将很快与苹果的用户见面，这也呼唤更多的开发者加入到Objective-C应用开发的行列中来，让这门本年度最佳编程语言为用户实现更大的价值。

高昂
中国标准化研究院助理研究员，从事信息技术标准化研究工作。关注开源社区，也是OSGeo中国和InfoQ中文站成员。

异构计算的挑战

进入新世纪之后，软件研发面临并行编程的技术变革、硬件架构面临异构计算的挑战，这些改变是否意味着新的机遇，取决于能否建立良好的生态链。

2004年12月，C++标准化委员会主席、著名程序员Herb Sutter在自己的个人网站发表了一篇影响深远的文章《免费午餐已经结束》（中文版发表于本刊2006年11月期）。它第一次使广大研发人员开始关注并行编程，也因此被JBoss之父Rickard Oberg称为该年度最重量级的技术文章。

这篇文章指出，硬件的突飞猛进，曾经使软件开发长

期以来一直在享用免费大餐：不用改变代码，只需更换新硬件，就能得到整体性能的提升。但进入新世纪之后，由于散热、功耗和电流泄漏等物理限制，通过不断提升时钟速度进一步提高芯片性能渐渐走入死胡同，除了继续扩大晶体管集成规模之外，要大幅提升性能，只能通过多核实现。软件研发面临一次重大技术变革——并行计算。

6年后，多核架构已经成为主流，甚至大部分高端移动设备里也都是双核处理器，超级计算能力进入家家户户甚至每个人的掌中。下一个硬件架构大趋势又将是什么？又将会对软件研发产生怎样的影响？

今年6月在西雅图举行的AMD Fusion Developer Summit上，“进入异构计算时代”成为事实上的主题：

AMD在大会期间的一系列发布，包括融合了Fusion A系列APU芯片、开放的FSA（Fusion System Architecure）架构、也将走向异构的下一代GPU、gDEBugger和PPA等多款开发工具等等，充分显示它全面拥抱异构计算时代的决心；

Herb Sutter在主题演讲中正式宣告继单核、多核之后异构计算时代已经到来，他还同时发布了基于DirectCompute的并行计算库C++ AMP，除了在下一版Visual Studio中提供支持外，C++ AMP也将作为标准向业界开放；

而在处理器市场举足轻重的ARM也在演讲中明确表示，认同异构计算是大势所趋，不顺应这一潮流的公司将被无情抛弃。

我们已经看到，无论是移动、桌面还是云计算各个平台，无论是在芯片级、机器级还是数据中心级，单位能耗所提供的性能、可扩展的并行处理能力等成为共同的焦点。在CPU的潜力挖掘渐尽的情况下，技术人员纷纷将目光投向天生适合并行计算的GPU。为了更好地支持HTML5在图形、多媒体上的表现，IE9、Chrome等现代浏览器都将GPU加速作为研发重点。在云端，Amazon EC2已经于去年年底增加了群集GPU实例，可以以极低的成本提供超级计算能力。AWS的灵魂人物之一James Hamilton此前甚至还讨论过包括FPGA、ASIC等更多不同指令架构的异构系统。而曾经夺得超级计算机世界冠军的天河1A，也配备了近万个GPU。

技术变革意味着挑战，更意味着机遇。芯片领域目前的格局变化已经充分证明了这一点。必须看到的是，并行计算其实并没有像多核硬件走入普通家庭那样在程序员社区普及。而与并行计算相比，异构计算对软件研发的挑战有过之而无不及，要进入主流可谓任重道远。这些都依赖于生态链的建设。标准的制定与推广，语言、编译器、框架、运行库等的支持，都不是易事。而开发人员的重视和培训，开发社区的建设和成功运营，更是重中之重。

作为目前唯一一家同时具有一流CPU和GPU设计能力的企业，AMD抓住异构计算的大趋势，首次举行面向开发人员的大会，开展奖金为5万美元的OpenCL编程大赛，推出开放的FSA架构（CPU和GPU中立）和Fusion战略争取更多的同盟军，已经走在了正确的道路上。

刘江
全球最大中文IT社区CSDN暨权威技术媒体《程序员》总编。北京图灵文化有限公司联合创始人、总编。曾任华章公司副总编、软件开发技术杂志Dr. Dobb's Journal中文版《软件研发》主编。

用数据库记录什么？

作为数据库的设计者，换一个思路，多花时间考虑如何扩展数据库应用领域，成功的几率也许会增加很多。

也许是因在这个行业呆得太久的缘故，一提起数据库我们禁不住的就会和企业应用、互联网站点联系在一起，但事实上数据库所能记录的远远多于这些，很多大型IT巨头也在用数据库（无论是关系数据库还是NoSQL数据库）记录我们的地球，但对IT行业而言，一般能激起广泛兴趣的往往和地理信息、空间信息有关。

日前，最大的全球植物数据库（TRY Global Plant Database）上线了，它用来记录植物功能特征或性状，其中科学家们已经将编制好的69万种植物的300多万种性状保存进去。这个数据库的意义除了研究全球植物变化外，更重要的是可以用它作为镜子，使我们了解气候变化、土壤开发究竟会对人类自身赖以生存的环境产生怎样的影响。用数据库记录我们的地球，让数据库成为我们保护环境的得力助手。

数据库只是工具和手段，但身在IT圈的我们很容易本

末倒置，为了数据库而数据库。数据库一度曾经是软件的中心，但在互联网热潮及随后Web 2.0风起云涌的时期，我们发现数据库逐步边缘化，数据库技术成了其他技术浪潮的陪客，直到最近大家发现传统集中式数据库存储成本和计算效率无法满足需要时，才开始NoSQL的革命。

但作为数据库的设计者，是否应该换个思路，是不是自己把路走“窄”了。现在，各家商业数据库厂商发布新产品的时候，提供的行业千篇一律——零售、医药、政府、电信……，尽管各家都声称自己在创新方面有了多大突破，但对比地看感觉连自己都没有突破。

尽管国内今年IT行业“年景”不错，数据库厂商凭借项目搭售预期收益还会快速增长，但我认为当前主要数据库厂商应该多花时间考虑如何扩展数据库应用领域，也就是用数据库记录什么的问题。过去的5年里，互联网新秀辈出，往往是当他们商业应用成功时业内才会发现原来记录并服务这些数据也会这么大的市场？

你也许会说找到这样的机会很难。确实。不过我们生活在一个完全“浸泡”在各类媒体的时代，新闻、报纸、网页中经常会流露出很多有价值的需求，如果有人关注这些需求、并作技术处理将他们量化后记录下来，成功的机会就会提高。

我们常说软件已经进入服务化、个性化时代，不仅电子产品更加强调个性化，就连应用软件也在强调个性化，但其实数据也许才是最具个性化的内容，表征不同关系、实体、内容的数据才是信息化服务中最关键的一项，洗尽铅华之后与用户切身利益关系最大的也是数据。

我们也常听到IT行业是需求推动的行业，不错，数据库技术更是如此，数据库技术的发展需要依靠数据库应用来推动，而个性化数据需求、便携数据使用要求、更加贴近生活和自然环境的数据需求也许是不错的切入点。

比如：我们记录员工信息的时候总会考虑到一些“八股”的字段，但如果放在矿山环境下，我们就应该为了生命救援需要，考虑记录他们的主要生命体征，如果放在工作频度很高的流水线环境下，为了减少次品率，可能需要记录员工们的各项情绪指数等。

总而言之，只有让数据库记录的内容不断创新，数据库自身才有可能快速进步，并可能重新回到软件中心的位置。数据库的价值不仅在于它服务应用的价值，关键在于它记录了什么、留下了什么，不是吗？

王翔
软件架构师，主要研究方向为XML、 E 、领域设计和 应用。工作之余喜爱旅游、写作和烹饪。

开放平台第二季，你准备好了吗？

腾讯的一举一动都颇受关注，Q+一推出就成为国内备受业界关注的项目，然而目标草根用户的Q+是否为这个市场做足了准备呢？

国内目前最令人期待的开放平台，莫过于Q+了。在Q+宣传页面的首页，有这么一句话：开放的互联网，你准备好了吗？我想，绝大部分程序员，还是没有准备好吧，包括Q+本身。因为Q+的主人马化腾，说过这么一句话：永不向360开放。

为啥大家那么关注Q+呢？因为QQ具有巨大的注册用户数。虽然，QQ到底有多少实际注册人数？这个数字，估计腾讯本身也不好统计（官方数字6.47亿），因为一个人实际可能拥有数个QQ号码，但是QQ最高在线用户，已经突破一亿四千万了，我们可以这么理解，这相当于中国的十分之一人口，曾经在同一个时间一起玩过同一款软件：QQ。这个数字，还大致相当于美国的一半人口。

腾讯向第三方开发商发出邀约，希望与大家一同构建互联网的未来（360除外）。据了解，“腾讯将尝试以API接口的形式通过Q+向第三方应用商提供如内容分享、文件传输、语音视频等核心功能组件，第三方应用商则可通过这个平台进行调用，将这些用户使用最多、最喜爱的核心功能植入到创新应用中”。

毫无疑问，第三方应用，也可以通过Q+API把自己的应用，像插件那样，植入到QQ或者Web QQ上，供6.47亿用户使用，这该是多么大的一个市场啊。

从开放平台成熟度上讲，目前，开放平台做得很成熟的

是淘宝，面向电子商务用户。面向后台的应用，第三方合作伙伴可以提供小到订单处理，订单打印等基本功能，中间层次的如：进销存软件，大的如ERP管理软件。面向前台的应用，第三方合作伙伴可以提供店铺装修，收藏有礼，淘大奖等等营销类应用软件，应该说，淘宝开放平台已经初步形成一个生态链。虽然位于生态链上的各个软件开发商也忒参差不齐了，大的软件供应商快撑死了，每天疲于奔命，应对客户不断变化的需求，小的软件商还在苦苦挣扎，可能一个付费的客户也没有。

像拍拍网，也做了开放平台，但跟淘宝开放平台的差距就跟拍拍与淘宝的差距一样，一个是原始社会，一个是资本主义初级阶段，没法比的。

从开放平台面向的客户群上讲，那么Q+呢？又回到了老话题：电子商务、移动互联网、云计算是未来的三个发展方向。淘宝是电子商务公司，实际客户是卖家，腾讯是互联网公司，实际客户是一个个社会草根，所以两家的开放平台是完全不同的思路。无论两个大佬是什么思路，作为草根的创业者，我们只有站在这些大佬的肩膀上，做一些创新型的产品，这就是我们的机会。因为Q+不可能主动为淘宝服务，淘宝也不可能为Q+写插件，360想直接跟Q+连起来，更不可能，但是我们第三方合作伙伴可以，这也是我一直主导的想法，所以我辞职创业了。

如果说：淘宝、腾讯、百度、新浪、网易、360、开心、人人纷纷开放自己的平台，算作第一季的话，那么这个故事如何开展下去？只有我们草根创业者参与进去，才能把这个市场做大、做活，才能进入到自由竞争的开放平台第二季，开放平台才能真正发展起来。

邢波涛
现任北京新软孚信息技术有限公司技术负责人。关注 管理软件和 2 、 2 电子商务的融合。

口令与隐私

不仅能够有效地保护公民的生命和财产，也能够有效保护他们的名誉和隐私，这样的信息社会才能够稳步向前发展。

就在我开始撰稿之时，英国女王正在二战盟军密码破译中心所在地向英国儿童宣布一项密码破解挑战赛。这是自美国总统奥巴马在电视讲话中提醒公众注意密码安全后，第二位西方国家元首就很细节的安全技巧直接与民众进行对话，给充满口令和密码迷局的本月带来了有趣的风景。

本月，微软决定强化Hotmail的口令安全管理，逐步禁止用户采用常见密码。很有趣的是，微软并没有采用所谓常见密码黑名单的策略，而是使用了统计策略，即微软认为使用人数高于一定阈值的密码就是不安全的密码。对于Hotmail这种大站来说，其密码原则上不应是明文存储，而是使用Hash，即密码禁忌问题变成了Hash频度统计问题。

索尼事件后，高频密码的统计结果被曝光。为了避免这种对具有大用户集的密码的破解，微软做出了一个非常巧妙的调整。

另一个与口令有关的事件是已经震惊全球的新闻集团“窃听门”。根据报道，此次事件中的很多短信窃听使用的口令竟然是通过猜测得到的。因为运营商会给用户设置一个简单的默认口令，而多数用户并不会自己修改这种毫无安全价值的口令，基于可能少数几个默认数字或者已知少量用户信息就可以尝试出大部分用户的口令。根据已经公开的的信息，其窃听手段无所不其极，包括在模拟电话时代通过信号扫描监听的方式窃听戴安娜王妃的手机。现在，尽管GSM和基站监听的成本也已经非常低廉，但通过默认密码来获取短信，确实是更加惠而不费的方法。每每有人为丢失手机而惴惴不安，却没有想到即使手机还在身边，通信的秘密也早已被窥视者洞悉。

我初涉信息安全时，一直以为默认口令的策略是初期运营商的愚蠢或者安全意识淡薄。但这些年过去了，他们依然没有改变。我们这才明白，这是以降低安全等级为代价来减少自己的支持成本。当用户因不知密码而无法操作某项业务时，客服会告诉你，如果您从未改过口令，可以尝试一下

0000、1234或者您有效证件的后四位。

我们很难将此完全归结为运营商的问题，毕竟多数用户更在意方便性，这始终是不可改变的事实。但运营商在安全方面的努力远远不够，也同样是事实。此外，这也向在RSA令牌召回后以为可以凭某些新技术和算法特点取代RSA的企业和组织发出了提醒——安全的最大阻力并非来自攻击者，而是来自传统。

本月，哈佛大学的一项研究遭到了批判，因为研究者下载了1700个Facebook用户的资料进行数据分析，经证实这些用户正好是哈佛2009届毕业生。研究人员因此被指控侵犯学生隐私。对僵尸和爬虫云集的中国微博、社交网站和通讯录来说，这种事似乎不算什么，但它却在美国却掀起了波澜。

随着云时代的到来和IT业寡头的形成，用户隐私保障越来越不取决于个人的安全能力和意识，而依赖于远端的云。但公众需要在与商业寡头的斗争中伸张自己的隐私权益，否则就会出现电影*America: From Freedom to Fascism*中那个有趣的场景——在你订一张披萨饼时，客服电话突然告诉你，翻开你媳妇订阅的某本杂志第几页有一张优惠卷可以使用。

在信息社会，公民不仅要能够有效地保护自己的生命和财产，也需要有效地保护自己的名誉和隐私。中国的信息安全产业能否稳步前进的一个重要支点，正是中国民众是否能建立起对个人隐私的崇尚、尊重和信仰。

肖新光
网名江海客，安天实验室首席技术架构师，研究方向为反病毒和计算机犯罪取证等。

Google+进行时

Google又一项社交尝试，似乎是有史以来反响最好的一次，在众多敌手环伺的竞技场上，搜索巨人的Google+最终能收获怎样的成绩？我们截取它从初生到现在的短短一个月时间的业界动态，也许能够窥测其未来走向。

2011年6月29日，作为Larry Page重登CEO之位后推出的重磅产品，Google+正式开放测试，此项目由Google高管Vic Gundotra和 Bradley Horowitz负责，令人耳目一新的用户界面则出自苹果前Mac软件团队设计师Andy Hertzfeld之手。

最初开发代号为“翡翠海”（Emerald Sea）的Google+，几乎涉及了Google公司所有的产品，数以百计的工程师参与了该计划，在Gundotra和Horowitz看来，Google+不是一个产品，也不是一种战略方案，而是一个扩展后的Google。事实上，Google+已经整合了Goolge所有的社交服务，诸如Google Profiles 和 Google Buzz，连旗下两项流行的服务品牌Picasa和Blogger也即将更名，作为Google+关键的组成部分，与其紧密集成。

六大功能

Google+的界面有六大功能，分别指向Stream、Photos、Circles、Profile、Huddle、Sparks和Hangouts。这其中，Stream是联系人的最新动态；Photos是照片功能，可以不断自动上传照片，甚至通过手机也可以即时上传。最重要的一个功能当属Circles，使用者可以选择和组织联系人，分成群，让分享最优化。Vic Gundotra就曾表示：“它是我们产品的核心。”

Profile是用户资料，据Google称，每一位用户的Profile必须开放搜索（可以选择部分开放），否则将会被删除。Huddle则是类似群组或多人聊天室，可与Android 整合。

Sparks被当做一种分享引擎，可以收集感兴趣的话题。Hangouts则提供多人视频聚会功能。

拥有上述六大功能，让我们看看接下来会发生什么。

首先，Google+放弃了通用消息发送模式，这是Facebook类社交网站的标配之一，但用户的信息可能需要被区别对待，因而，Google+允许用户自己建立不同的Circles，并且可以在不同的“社交圈”里发布不同的信息，跟合适的人做合适的事。在美国著名IT杂志PCWorld网络版刊登的文章里，作者Preston Gralla称，这将是Google+有望击败Facebook最重要的因素。

除了可以定向信息流动，Circles另外一个优点是对用户隐私更好的保护。New York Time撰稿人Nick Bilton就用自己的亲身经历证明了这一点，作者发现帖子是默认设置为私人

的。Google+ 的用户需要自主操作，以使他们发布的内容公开。Google这样做可以使用户免于公开的危险。相比之下，Facebook等社交服务商则常常在用户没有注册新功能的情况下就得到了用户信息，并且“需要使用像迷宫一样的按钮和菜单，才能够让他们的个人信息保持私密性”。“以往用户继续使用其服务，说没有可行的替代性社会网络产品，和家人朋友进行交流”。但当Google提供最好的隐私保护服务时，以后的状况就会截然不同了。

其次，Google搜索引擎将为Sparks提供强大的支持，可以向用户提供博客发帖、新闻、视频以及用户感兴趣的任何内容。当要与其他人分享信息时，只要轻点分享键，用户就可以与其“社交圈”进行共同分享。这将极大地激发用户交流兴趣。

第三，Google+整合了Google其他服务，譬如在Google+上的谈话内容会显示在用户的Gmail里。Google+上还提供Google Talk的直接链接。用户还可以期待更多的兼容服务。

第四，Hangout将提供一个具有潜在增长能力的领域，商业应用。最多支持10人的电话会议，可以为管理企业、快速召开在线会议提供良好的途径。实际上，它可以被看作是工作工具而不仅仅是社交工具。

最后，移动功能强大，与Facebook不同，Google+从一开始就充分考虑到了移动功能。因此，Google+更适合在智能手机上运行（已经推出了相应的Android程序，与苹果达成罕见的合作也让Google+在Apple Store上线）。比如，用户可以使用“Hangouts”功能群发消息。上载照片和地理位置标签也能够轻松实现。只要你的Google+上有任何新活动，用户就会收到发送到手机上的文本信息。尽管Facebook也能够实现一些类似的功能，但Google+却可以比Facebook做得更好。

反响与评价

Google+乍一推出即受到热捧，用户量迅速膨胀，迫使Goolge不得不关闭邀请注册，投资者对于Google+的看法也好过预期，新产品提振了Google近来疲软的股价（今年以来已经下跌近20%）。

Google随即在7月7日晚重新开放注册，受到用户的强烈欢迎，Google+工程总监Dave Besbris在宣布重新开放邀请注册时说：“目前系统运行良好，因此我们判定可以短暂开放邀请注册。我们的目标是将公测用户量提高一倍。”此时Ebay上兜售的Google+邀请码已被炒到每个75美元。

从推出到宣布用户数量突破1000万，Google+仅用了不到半个月的时间，同时，Google+用户的每天分享也达到10亿次。这个增长速度在互联网社交服务历史上从未有过，甚至也超出了Google的预想。Google+的搜索也呈现出爆炸式增长，据统计，Google+在Google的搜索结果已经超过10亿条。有人形容Google+引发了社交网络的地震，“Google+就是把Facebook、Twitter、甚至新浪微博的各种功能加到一起”。著名IT博客Tristan Louis认为Google+的出现使得“Facebook面临围剿”。

岂止是Facebook，LinkedIn、Twitter被都拿来与Google+进行比较。

对手Lin ed n

7月6日，福布斯杂志网络版撰文称，Google+将对LinkedIn形成冲击。相比其他社交网站，Google+的优势在于凭借着在搜索引擎市场的强势话语权、丰富的个人资料和身份信息选项，用户在Google+可以对自身形象有效控制，进而拥有更强大的个人品牌，这是职场人士会去使用Google+的重要原因。而LinkedIn显然不具备这方面的优势，并且在帮助用户建立联系方面表现较弱。

对与外界的评论，LinkedIn CEO Jeff Weiner称，Google+是在利用用户空闲时间的需求，而用户没有这个空闲时间，“你看不到人们在使用Twitter的同时在使用Facebook，或者在使用Facebook的同时使用LinkedIn”。并且由于不能与流行的三大社交网络融为一体，Google+将处于困境之中。

对手Twitter

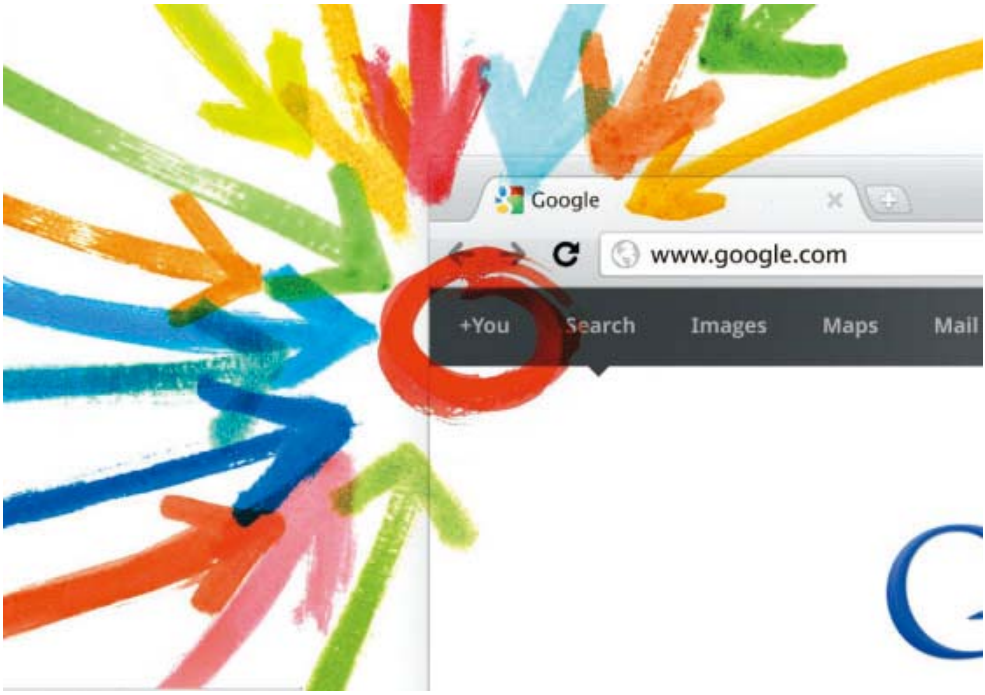
在推出Google+之前，Google还有一个失败了的微博服务：Buzz。Bradley Horowitz说，“在Buzz上我们学到很多，其中一点就是：一款产品真正的市场机会在于解决用户的隐私担忧，以及信息如何分享。”

不但从自身总结教训，也从他人成功里汲取经验，有人认为Google+很像是Twitter的进化版，甚至包括了特殊状态更新。就在推出Google+后，Google与Twitter迅速分道扬镳，双方续签实时数据许可协议失败，Google发言人就此表示，“我们计划在实时搜索中整合Google+和其它社交网络的信息。”

与此同时，微软希望与Twitter续签一项长期协议，Bing将有望同时获得Twitter公司和Facebook公司的实时数据。据Twitter表示，除此之外，它还与诸如雅虎、NTT Docomo、雅

虎日本以及其他数十家小型开发商签订有数据许可协议。

Google十分清楚，他们的成功主要依靠两大领域：搜索和在线广告。Twitter正在逐渐成为他们的第一大竞争对手，传言每年广告收入可达6000万～7000万美元，Google自然按捺不住，决定见招拆招。由此可见，Google+带动了新一轮搜索数据的争夺战，与Twitter的正面交锋也刚刚拉开帷幕。



对手 ace oo

还看不出来Google+会给Facebook带来如何沉重的打击。Facebook拥有7亿用户远超Google+目前的几千万。5月，美国访问者平均每人花375分钟在Facebook上，而Google只有231分钟。但双方的往来交锋早已“默契”进行。Google+推出后不久，7月6日，Facebook CEO Mark Zuckerberg宣布将与刚刚被微软纳入麾下的Skype合作推出视频聊天服务，分析认为，此举意在与Google+，特别是其Hangouts功能，点对点竞争。

Facebook的众多高管集体亮相Google+，其中包括COO Sheryl Sandberg，CTO Bret Taylor以及CEO Mark Zuckerberg。和Twitter不同，Google不会核实用户帐户的真实身份，除非用户自行举报，所以当Mark Zuckerberg真身曝光时，他也颇为不解，“为什么人们会对于我拥有一个Google+账户感到如此惊奇？”，看来Facebook相当看重这样一位对手。

这样的对手的确相当可怕。Google将把从搜索到视频（YouTube）几乎所有在线服务整合到Google+中。这样使用户可以在Google+上进行更新、留言、分享等，数以千万计的用户在使用Google的服务（Gmail、GReader、GDocs等），Google+与这些服务都有紧密联系，这极大地增强了Google+的用户黏度，用户离开Google+去其他社交网站的可能性非常小，但却成为Facebook转向Google+的最大理由。

当Mashable主编Ben Parr直接向Zuckerberg提出如何看待Google+时，后者给出了自己的几点看法。

一是针对Facebook自身定位的，“与寻求多元化发展的公司相比，专注于某一个领域发展的公司更容易在该领域取得成功。”

二是评价Google+的，Circles能够让用户把其他人邀请进特殊的群组，但他们并不知道自己所在的具体群组以及这些群组中的其他成员。总体上说，这一过程既耗时又缺乏关注度。

归根结底，真正需要的还是用户基础，这恰恰是Facebook所拥有的，可却是Google+目前绝对不具备的竞争优势。

未来与结语

Google+的受欢迎程度无疑让Google工程师兴奋不已，新的功能与整合也在迅速迭代发布，截止发稿前，Google高级副总裁Vic Gundotra宣布即将开放Google+ API给第三方开发者。

Google在社交产品上的努力可谓屡战屡败，屡败屡战，虽然很多次信心满满地发布，加上一贯的邀请码机制，使得发布伊始，使用者趋之若鹜，但之后很快就被人遗忘，类似“雷声大雨点小”的例子我们可以举出一连串，自2004年开始的Orkut、OpenSocial、FriendConnect、Lively，到最近要么放弃要么冷落的Google Wave、Google Buzz、Aardvark。

Google一直致力将互联网中海量无序的数据，用技术的手段整合管理，但人的身份与关系是否能被规整到一个个Circles中呢，至少现在妄言Google+成败，为时过早。

不妨引用Zuckerberg的一句话作为结语，“如果Facebook能够开发出最好的服务，那么就会实现更高的价值。如果Facebook无法做到这一点，那么就会被别人超越”。这也同样适用于Google+。

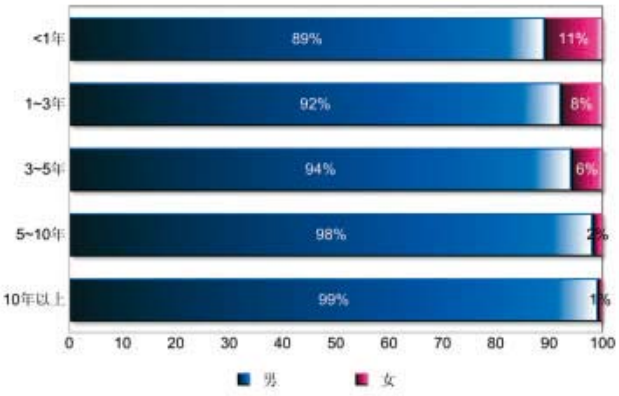
20 中国软件开发者年度调查

记者 常政

由CSDN和《程序员》杂志举办的“中国软件开发者年度调查”，已经历时7个年头。这段时间内，世界在变、产业在变，开发者也在变，而我们的调查无论是调研方式、数据采集渠道，还是样本处理方法等也顺应时代和技术发展，进行了相应革新。本次年度调查广泛涉及开发者社区、技术、工具、厂商等各个层面，除了延续以往的规范化、全面深入的特性，今年最大的特性是根据2010~2011年的技术趋势和热点，以及厂商新推出的产品技术，对调查项进行了调整和增补，力求准确及时地反应中国软件市场的最新变化。由于篇幅所限，本文仅管仲窥豹，捡其要点，以飨诸位读者。

软件开发者：孤独而稳定的群体

不同从业时长的软件开发者性别分布

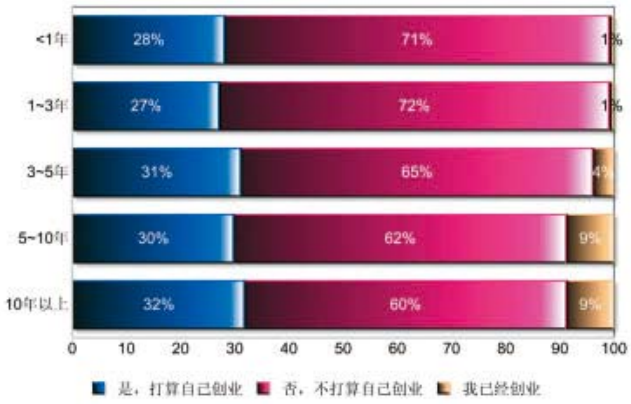


样本量 N=2,962 数据来源：2011年中国软件开发者年度调查

从性别比例看，开发者真的是孤独的群体，从事1年以上的开发者当中，男性开发者均占据90%以上，而且从业时间越长，女性比例越低。看来目前的中国软件，不具备让女性开发者茁壮成长的肥沃土壤。从学历分布看，各个从业时长的开发者基本50%以上都是本科毕业。近年来，随着外包软件基地在国内的纷纷创建，以及软件职业技术教育的繁荣，我们可以看到从业时长较短的开发者中，专科/本科学历的比例在扩大，高中/中专学历的开发者的比例也在提高。然而随着时间的延长，低学历的开发者比例陡然降低。看来软件开发本质是一项智力创造，要想持续发展，一定程度的教育是必需的。

软件开发总体来说是一项稳定的职业，和我们预料的一样，收入和职位高低与从业时间呈正比例相关，一般3年以上都能超过月薪5000元。大多数开发者并没有创业打算，说明总得来说，其收入水准并没有给他们的生活带来太大困扰。而对于开发者创业，最显著的槛是3~5年，这段时间内创业比例陡然上升，以后趋于平缓。看来“5年”是很多开发者确定事业

不同从业时长的软件开发者的创业计划分布



样本量 N=2,962 数据来源：2011年中国软件开发者年度调查

定位的时间。

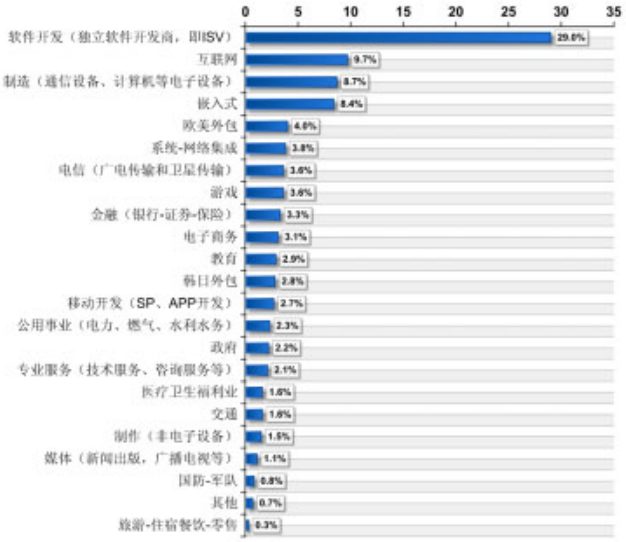
中国软件：产业格局潜流暗涌

对于中国软件开发者所在的企业，29.0%的软件开发者所在企业是软件开发行业，其次是互联网和制造业，分别占9.7%和8.7%。这样的数据和往年调查情况类似，说明中国软件业的产业链基本格局没有太大变化。

在开发者主要开发的软件类型中，比例最高的是互联网开发，前、后端累计28%。曾长期占主导地位的桌面客户端应用（含C/S架构）退居第二，为20.3%。异军突起的是嵌入式应用/工业控制系统，比例为16.3%，在物联网时代到来的大背景下，它的增长势必会逐年持续。移动应用的比例也引人关注是9.7%。企业级应用是14.4%，这个比例不高，看来在时代新潮流的冲击下，传统软件开发的领地正逐渐缩减。

总的来说，中国软件的整体产业链格局，尽管和往年比并没有太大变动，但是潜流暗涌，各个领地的力量对比正在悄悄发生变化——中国软件正走在一条全新的转型之路上。

中国软件开发者所在公司行业分布



样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

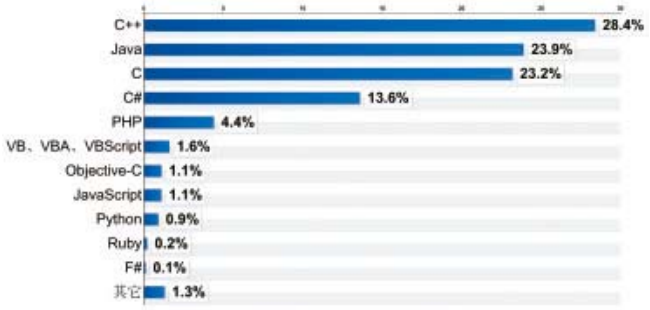
中国软件开发者主要开发的软件类型分布

样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

编程语言：Java和 C/C++各领风骚

本次调查发现，不同从业时长的开发者使用的第一编程语言存在差异，从变化来看，使用Java、C语言的开发者在新开发者中比例比老开发者更高，而C++和C#的使用率在从业时长较久的开发者那里使用率更高。尽管如今层出不穷的编程语言/平台颇有一番“乱花渐欲迷人眼”之势，但Java和C/C++这类经典的编程语言，凭借累年的工业积累和社群支持，如同一坛老酒，历久弥新，新手们选择这些语言切入IT 行业不失明智之举。PHP等Web动态语言的使用比例普遍比Java和C/C++低，不能因此断言它们的衰弱，因为在中国软件业内，Web 开发领域毕竟整体从业人数仅10%左右，远低传统软件开发领域。在所有的脚本语言中，JavaScript一枝独秀，在10年内的各个从业时长的开发者中均占据了20%以上。JavaScript刚出道时，是作为给非程序员用的脚本语言来推广的，但随着Web

中国软件开发者使用的第一编程语言分布



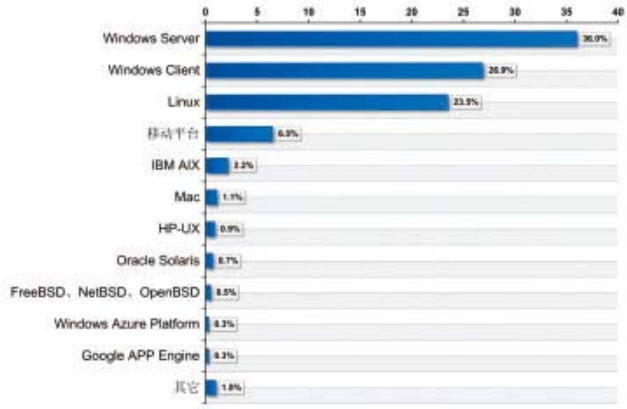
样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

2.0、RIA的迅速普及，JavaScript逐渐在开发者们心中占据越来越重要的位置，而像jQuery、MooTools、Prototype等JavaScript框架和类库的出现，更是令它如虎添翼。

再来看操作系统，没有意外的是，开发者在项目中使用Windows Server和Windows Client所占比例最高，分别为36.0%和26.9%，其次是Linux和移动平台，分别为23.5%和6.5%。Linux 尽管近两年拥有了移动平台这个可共同抗衡微软的同盟军，但Windows 系列占据开发者们所选系统半壁江山的局面还是很难改变。

数据库方面，中国软件开发者主要使用的数据库中，SQL Server、Oracle、MySQL呈三足鼎立之势，比例分别为23.0%、29.5%和28.3%。这三款数据库在易用性、安全与性能、价格上各有千秋，对应不同类型的用户。目前开发的数据库应用里，大多数是规模小于10万条记录的，其开发者比例在57.4%，而10万～100万条记录的比例在22.3%，100万～1000万条记录的开发者比例为12.5%，大于1000万的比例是7.7%。结合使用的数据库品牌和应用对数据记录的实际需求两个指标综合考虑，

中国软件开发者目前开发的项目所应用的操作系统分布



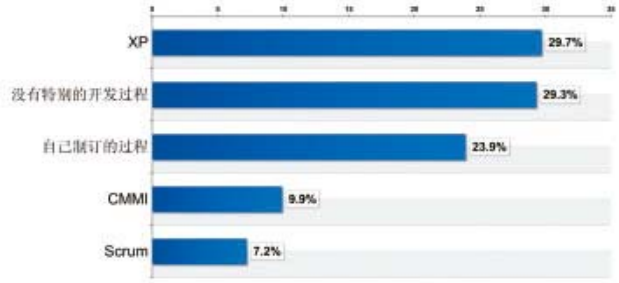
样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

可以发现开发者相对更注重数据库的性能。随着Web 2.0、SNS的兴起，我们发现非关系型数据库NoSQL也越来越受到到开发者们的关注，据调查MongoDB、Cassandra是最受大家热捧的NoSQL，分别占26.1%、20.1%。

软件工程：XP 一枝独秀

在软件工程里，绝大多数开发者都采用敏捷模式，两种方法论：XP（29.7%）、Scrum（7.2%）累计占据了37%的比例。XP和Scrum，都体现了快速反馈，强调交流，强调人的主观能动性等敏捷基本原则，区别在于XP更注重强有力的工程实践约束，而Scrum则突出Self-Orgnization（管理），目前看来XP在开发者中应用更加广泛。此外，23.9%的开发者所在公司自己定制过程，而使用CMMI的开发者比例为9.9%。总体来说，接近70%的开发者们按照定制的过程进行开发，从中可一窥中国软件工程规范化、标准化水准的现状。在研发管理工具方面，大多数开发者使用公司自行开发的工具，占据35.8%，而应用最多的两个品牌产品是Microsoft VSTS和IBM Rational，比例分别为占23.8%和21.9%。

中国软件开发者软件开发过程统计



样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

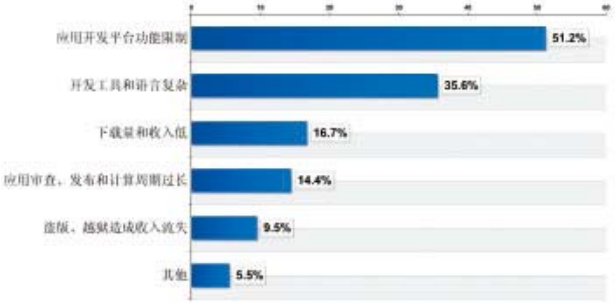
移动应用：方兴未艾

移动应用尽管在中国软件生态链格局里占据并不多的份额，但绝对在开发者心目中占据重要位置，它不仅预示产业和技术的未来潮流，而且随着几年前苹果的App Store 模式的推出，使得开发者的创业梦想成为无穷可能，凭一款应用而日掘千金已经被证明不是神话。据本次调查显示，在中国的开发者中，约有20.5%从事过移动开发，这其中65.4%从事这个行业不足一年，同时高达93%的开发者都是在三年以内。业界常把2007年iOS/Android的问世、2008年苹果App Store的推出，看做移动互联网发展的里程碑事件，现在看来并不为过。从团队规模看，目前中国的移动开发基本以个人和小团队为主，个人开发者占了近3成，2～5人的开发者为33.2%，6～10人的开发者占17.3%。表面看移动应用似乎成为个人或者小团队创业的福音，但从进一步的数据看，情况并不乐观，目前52.4%的移动开发者没有收入，仅有8.9%的人对盈利状况感到满意。

当然就移动开发者群体的主观感受而言，收入匮乏并不是最值得担心的，毕竟现在刚刚进入移动互联网时代，大量潜力还没有得到充分挖掘，所以一段时间内的清贫都是可以承受，目前开发者们最关心的还是如何“多、快、好、省”地做出满足用户需要的产品，因此最困扰大家的难题是“应用开发平台的功能限制”，占据51.2%，排第二位的是“开发工具和语言复杂”，有35.6%。

就移动应用的具体内容来看，中国移动开发者过去一年

中国软件开发者移动应用开发遇到的最大问题分布



样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

开发过的移动应用主要是手机单机游戏和手机小工具，比例分别为48.8%和30.8%，其次是手机网络游戏和社会化应用，比例分别为18.6%和11.0%。而在未来一年的开发计划里，手机单机游戏、手机小工具的开发比例下降，分别为34.0%和25.4%，企业/行业应用、社会化应用等应用的开发比例明显上升，反映

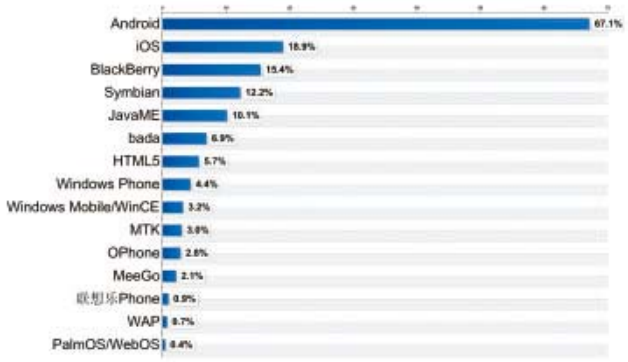
中国软件开发者未来一年计划开发的应用类型分布

样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

出开发者对这些应用正寄予厚望。

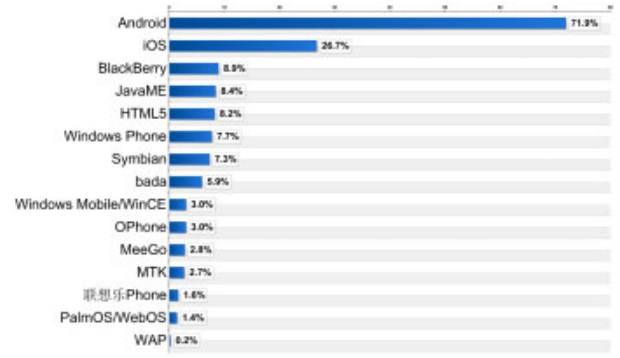
对于移动创业者，如何在五花八门的移动平台里选择合

中国软件开发者过去一年移动应用所针对的平台分布



样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

中国软件开发者未来一年移动应用所针对的平台分布



样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

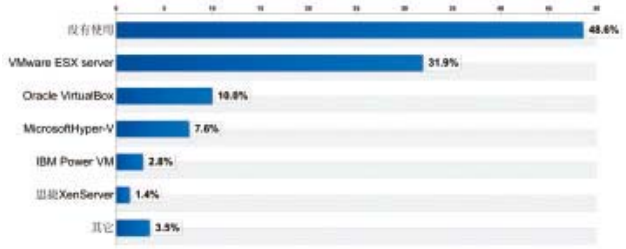
适的切入，是必须面对的抉择。从调查看，Android显得众望所归，71.9%的开发者将在下一年选择它开发应用程序，排名第二的iOS则占据26.7%的比例。

云计算：亟待进一步落地

云计算，和移动开发、物联网一样，也是业界公认的未来重要发展趋势之一。近两年来，业界上下对于云计算的炒作不遗余力，各大城市的云基地也纷纷破土而出，颇有一幅“大风起兮云飞扬”般的兴盛气象。然而根据我们这次调查，发现云计算离真正落地中国还有一段距离，目前48.6%的开发者还没有使用过任何虚拟化产品。

在云计算平台使用状况的调查中，从没有使用的开发者占62.1%，而平台使用率最高的是新浪和谷歌Apps Engine，分别为12.4%和9.6%。尽管云计算产品和技术使用率不高，但开

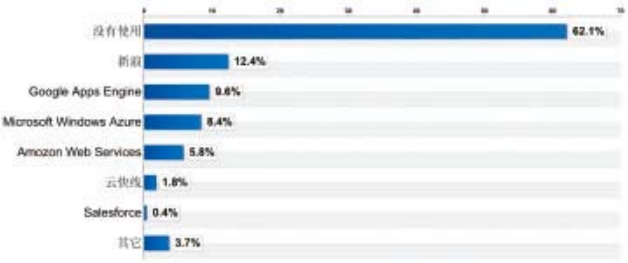
中国软件开发者使用过的虚拟化产品分布



样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

发者们还是给予了极大的关注，在“开发者最关注的技术方向”调查中，云计算以38.9%高居第二位（第一是移动开发，占48%）。和移动开发不同，云计算无论是技术门槛，还是创业成本都很高，再加上牵扯一系列法律监管问题，目前只有少数能调动各方面资源的IT巨头或者资本权贵才玩得起，大多数开发者驻足观望也不足为奇，云计算在中国的进一步落地，还需要些时间。

中国软件开发者使用过的云计算平台分布



样本量 N=2,962　　数据来源：2011年中国软件开发者年度调查

结束语

看完“2011中国软件开发者年度调查”的各种数据，我最想说的一句话是“青草正在悄悄生长”。从整个产业格局，从定性的层面，中国软件以及开发者群体的生存状态，没有太大的变动，但微观到具体的应用领域，在产业大趋势（移动、云计算等）的冲击下，你会发现各种力量（产品、技术、市场）对比正此消彼长，新的产业链正逐渐形成——中国软件正踏步在转型的大道上，而对于希望进一步提升自己的开发者来说，正迎来了未雨绸缪、蓄势待发的关键时机。

本文引用的数据，仅是“2011中国软件开发者年度调查”的极小一部分，如欲看完整的调查报告，请注意关注CSDN网站。

海量数据

策划 / 本刊编辑部

随着互联网、移动互联网和物联网的发展，各种终端、信息收集器的数量和种类不断增加，我们每个人、世间万事万物每时每刻产生的大量数据，都在不断进入信息系统，等待存储、分析和充分利用。我们将这些浩如烟海的数据统称为海量数据，也就是国外人们更喜欢说的Big Data。

数据再多，本身并不具有价值，只有将数据合理存储下来，经过清理、过滤和挖掘，并将分析结果以各种直观的方式呈现出来，或者实时地提供服务，提升用户体验，才能体现出价值，创造实际效益。必须指出的是，海量数据绝非巨型企业的专利，而是成为越来越多企业面临的问题。今年4月CSDN云计算调查显示，超过50%的受访企业每日生成的数据量在1TB以上，超过10TB的有10%，有5%的企业每日生成的数据量已经达到了50TB以上。某种意义上，处理海量数据的能力已经成为企业之间竞争的焦点，甚至与企业的生死存亡息息相关。

然而，今天的海量数据规模之大远远超过人们的直觉，这样规模数据的存储和处理，需要采用许多新的技术和思考方式，无疑给IT技术人员提出了巨大挑战。

本期封面报道中，将有来自支付宝、百度、Yahoo!、SAP、Teradata、新浪、淘宝、麦包包、优酷、Admaster、Esri等业内领先的软件企业和互联网企业的专家，分享他们在应对海量数据挑战方面的见解、经验和技术实战。内容涵盖运维、NoSQL、C++MapReduce框架HCE、数据魔方平台架构、电子商务推荐引擎、视频网站Big Data实践、Hadoop多维分析架构、海量空间数据库、《纽约时报》的数据可视化实践等。


ЧИТА

i t 技术综述

■ 文 / 蒋杰

Big Data是近来的一个技术热点，但从名字就能判断它并不是什么新词。毕竟，大是一个相对概念。历史上，数据库、数据仓库、数据集市等信息管理领域的技术，很大程度上也是为了解决大规模数据的问题。被誉为数据仓库之父的Bill Inmon早在20世纪90年代就经常将Big Data挂在嘴边了。

然而，Big Data作为一个专有名词成为热点，主要应归功于近年来互联网、云计算、移动和物联网的迅猛发展。无所不在的移动设备、RFID、无线传感器每分每秒都在产生数据，数以亿计用户的互联网服务时时刻刻在产生巨量的交互……要处理的数据量实在是太大、增长太快了，而业务需求和竞争压力对数据处理的实时性、有效性又提出了更高要求，传统的常规技术手段根本无法应付。

在这种情况下，技术人员纷纷研发和采用了一批新技术，主要包括分布式缓存、基于MPP的分布式数据库、分布式文件系统、各种NoSQL分布式存储方案等。

10年前，Eric Brewer提出著名的CAP定理，指出：一个分布式系统不可能满足一致性、可用性和分区容忍性这三个需求，最多只能同时满足两个。系统的关注点不同，采用的策略也不一样。只有真正理解了系统的需求，才有可能利用好CAP定理。

架构师一般有两个方向来利用CAP理论。

■ Key-Value存储，如Amazon Dynamo等，可以根据CAP理论灵活选择不同倾向的数据库产品。

■ 领域模型+分布式缓存+存储，可根据CAP理论结合自己的项目定制灵活的分布式方案，但难度较高。

对大型网站，可用性与分区容忍性优先级要高于数据一致性，一般会尽量朝着A、P的方向设计，然后通过其他手段保证对于一致性的商务需求。架构设计师不要将精力浪费在如何设计能满足三者的完美分布式系统，而应该懂得取舍。

不同的数据对一致性的要求是不同的。SNS网站可以容忍相对较长时间的不一致，而不影响交易和用户体验；而像支付宝这样的交易和账务数据则是非常敏感的，通常不能容忍超过秒级的不一致。

ac e篇

缓存在Web开发中运用越来越广泛，memcached是danga.com（运营LiveJournal的技术团队）开发的一套分布式内存对象缓存系统，用于在动态系统中减少数据库负载，提升性能。

memcached具有以下特点：协议简单；基于libevent的事件处理；内置内存存储方式；memcached不互相通信的分布式。

memcached处理的原子是每一个（Key，Value）对（以下简称KV对），Key会通过一个hash算法转化成hash-Key，便于查找、对比以及做到尽可能的散列。同时，memcached用的是一个二级散列，通过一张大hash表来维护。

memcached由两个核心组件组成：服务端（ms）和客户端（mc），在一个memcached的查询中，ms先通过计算Key的hash值来确定KV对所处在的ms位置。当ms确定后，mc就会发送一个查询请求给对应的ms，让它来查找确切的数据。因为这之间没有交互以及多播协议，所以memcached交互带给网络的影响是最小化的。

MemcacheDB是一个分布式、Key-Value形式的持久存储系统。它不是一个缓存组件，而是一个基于对象存取的、可靠的、快速的持久存储引擎。协议与memcached一致（不完整），所以很多memcached客户端都可以跟它连接。MemcacheDB采用Berkeley DB作为持久存储组

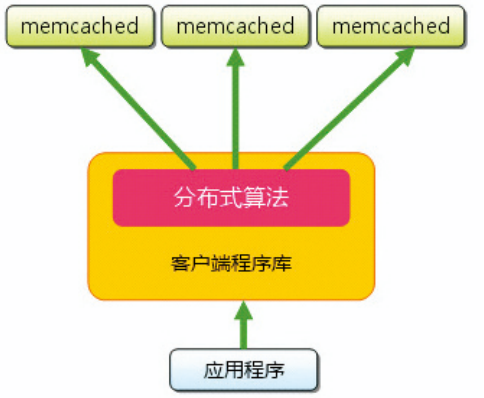


图1　memcached构成

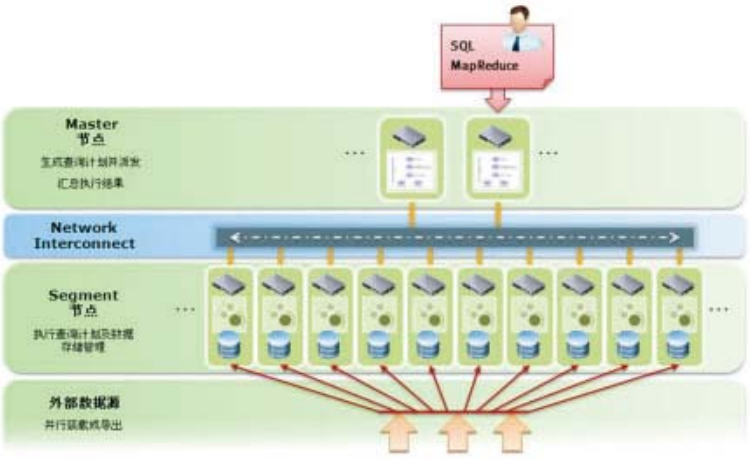


图2　Greenplum数据引擎软件

件，因此很多Berkeley DB的特性它都支持。

类似这样的产品也很多，如淘宝Tair就是Key-Value结构存储，在淘宝得到了广泛使用。后来Tair也做了一个持久化版本，思路基本与新浪MemcacheDB一致。

分布式数据库篇

支付宝公司在国内最早使用Greenplum数据库，将数据仓库从原来的Oracle RAC平台迁移到Greenplum集群。Greenplum强大的计算能力用来支持支付宝日益发展的业务需求。

Greenplum数据引擎软件专为新一代数据仓库所需的大规模数据和复杂查询功能所设计，基于MPP（海量并行处理）和Shared-Nothing（完全无共享）架构，基于开源软件和x86商用硬件设计（性价比更高）。

分布式文件系统篇

谈到分布式文件系统，不得不提的是Google的GFS。基于大量安装有Linux操作系统的普通PC构成的集群系统，整个集群系统由一台Master（通常有几台备份）和若干台TrunkServer构成。GFS中文件备份成固定大小的Trunk分别存储在不同的TrunkServer上，每个Trunk有多份（通常为3份）拷贝，也存储在不同的TrunkServer上。Master负责维护GFS中的 Metadata，即文件名及其Trunk信息。客户端先从Master上得到文件的Metadata，根据要读取的数据在文件中的位置与相应的TrunkServer通信，获取文件数据。

在Google的论文发表后，就诞生了Hadoop。截至今日，Hadoop被很多中国最大互联网公司所追捧，百度的搜索日志分析，腾讯、淘宝和支付宝的数据仓库都可以看到Hadoop的身影。

Hadoop具备低廉的硬件成本、开源的软件体系、较强的灵活性、允许用户自己修改代码等特点，同时能支持海量数据存储和计算任务。

Hive是一个基于Hadoop的数据仓库平台，将转化为相应的MapReduce程序基于Hadoop执行。通过Hive，开发人员可以方便地进行ETL开发。

如图3所示，引用一张Facebook工程师做的Hive和Hadoop的关系图。

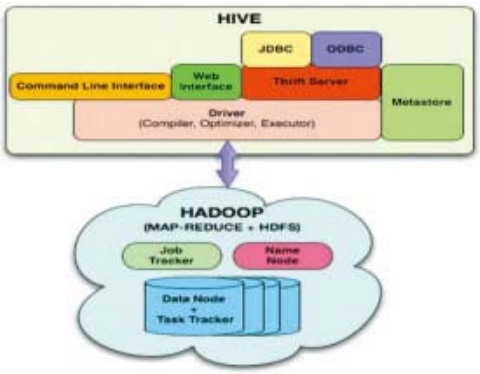


图3　引自Facebook工程师的Hive与Hadoop关系图

oS　L篇

随着数据量增长，越来越多的人关注NoSQL，特别是2010年下半年，Facebook选择HBase来做实时消息存储系统，替换原来开发的Cassandra系统。这使得很多人开始关注HBase。

Facebook选择HBase是基于短期小批量临时数据和长期增长的很少被访问到的数据这两个需求来考虑的。

HBase是一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，利用HBase技术可在廉价PC Server上搭建大规模结构化存储集群。HBase是BigTable的开源实现，使用HDFS作为其文件存储系统。Google运行MapReduce来处理BigTable中的海量数据，HBase同样利用MapReduce来处理HBase中的海量数据；BigTable利用Chubby作为协同服务，HBase则利用Zookeeper作为对应。

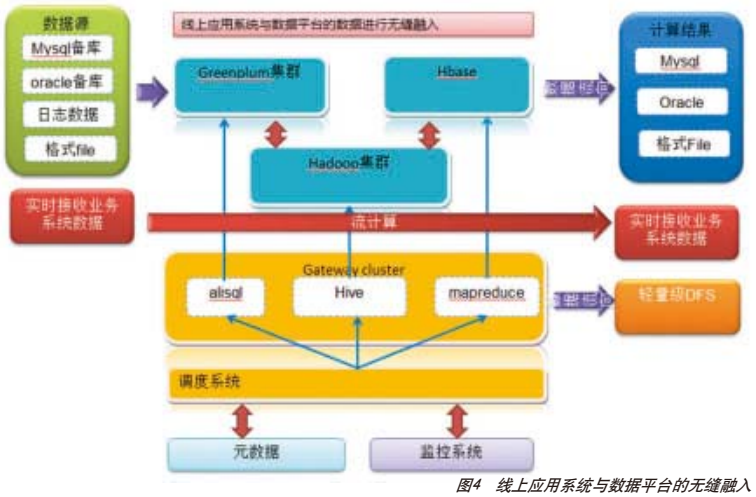


图4 线上应用系统与数据平台的无缝融入

总结篇

近来NoSQL数据库的使用越来越普及，几乎所有的大型互联网公司都在这个领域进行着实践和探索。在享受了这类数据库与生俱来的扩展性、容错性、高读写吞吐外（尽管各主流NoSQL仍在不断完善中），越来越多的实际需求把人们带到了NoSQL并不擅长的其他领域，比如搜索、准实时统计分析、简单事务等。实践中一般会在NoSQL的外围组合一些其他技术形成一个整体解决方案。

准实时的统计分析

■ 传输时统计分析，Stream Processing技术：FlumeBase、S4。

FlumeBase：可参考 http://flumebase.org/documentation/0.1.0/UserGuide.html中的quick start和architecture两部分。

S4：Yahoo!开源数据来流计算实时框架，可参考http://labs.yahoo.com/files/KDCloud%202010%20S4.pdf。

■ 查询时统计分析，结果集较小时，可以直接在返回前做统计分析处理。

比如买家消费记录查询的HBase实现，Schema设计，rowkey=uid，column=搜索词和查询值，version=交易id。

搜索相关

■ 充分利用NoSQL（比如HBase）内部数据的有序性、Row Key、Column Family、Version Timestamp。

我们用“HBase+二次索引”来实现实时营销的解决方案。也可以参考Facebook Message的解决方案：http://blog.bluedavy.com/?p=258。

■ 构建一个外围系统完成索引建立。

Google MegaStore：原文链接为http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper32.pdf，中文译文链接为http://cloud.csdn.net/a/20110216/291968.html。

也有人开始尝试基于HBase的MegaStore实现，链接为https://github.com/drevell/megalon。

简单事务

■ 事务处理服务 + NoSQL存储。

■ 淘宝开发的千亿级海量数据库Oceanbase，通过update server角色执行将写操作限制在一台机器上，实现事务。参考链接为：http://www.nosqlnotes.net/archives/170。

■ Google MegaStore通过为用户提供机制，根据应用特点划分entity group，将事务涉及的数据分布到一台机器上，实现事务。

蒋杰

花名平原君，支付宝数据平台基础架构负责人，主要研究方向为分布式数据库、行业数据建模、高并发查询系统架构设计、实时数据仓库。个人博客为 tt i ot t。

圆桌论坛：如何应对海量数据的挑战

■ 记者 / 董世晓

海量数据的浪潮有多迅猛？IDC在2006年估计全世界产生的数据量是0.18ZB（1ZB=100万PB），而今年这个数字已经提升了一个数量级，达到1.8ZB，差不多对应全世界每个人一块100多GB的硬盘。这种增长还在加速，预计2015年将达到近8ZB。目前IT系统的存储能力远远不足，就更不用说深入地挖掘和分析了。

在本文中，百度首席科学家**威廉 张**、Teradata首席客户官**周俊凌**、Yahoo!北京全球软件研发中心架构师**韩轶平**、SAP中国区企业信息管理咨询资深顾问**杜韬**等四位业内专家，将分享他们在应对海量数据挑战方面的见解和经验。

您所在企业的数据量现在达到了什么规模？

威廉·张：这个问题比较容易回答。百度不是一个产品，不仅有搜索引擎，还包括很多社区产品和媒体产品，所以这个数字大概是数百个PB，每天处理的数据大约有几十个PB。我是差不多四年半前加入百度的，所以我比较清楚地记得那时候的规模。与那时相比，现在的数据规模成长比较惊人，大概是那时的500~1000倍。

数据量大并不可怕，问题是要实时处理数据，因为任何的时延都会使服务失去一些优势，从而导致商业经济的下降。我们所做的策略都是针对实时性的，而且今天互联网用户的需求更加实时化，比如说微博、团购、秒杀。

周俊凌：从IDC的数据统计报告来看，数据增长是非常快的。相对于具体的数据量，Teradata更关注数据发展的趋势，并大量投入研究这种发展趋势，包括BI方面的变化和增长模式，这个模式对于我们非常有价值，通过研究这种模式，包括每分钟、每秒钟交易量有多大等这些数据的发掘和建模，数据科学家进行研究和探讨，把这些技术应用到生产系统里面，对企业发挥作用。

韩轶平：Yahoo!的主要云计算平台Hadoop现在有34个集群，总数超过3万台机器，最大的集群是4000台左右，总存储容量超过100PB。这个数量级可以说并不大，主要原因在于我们最近将很多精力放在处理用户隐私性和数据安全性上，因为按照欧盟的规定，Yahoo!不能存储超过一年的数据，所以我们的应对措施就是：不保存原始数据，但做很深入的数据挖掘，挖掘出真正蕴含的有价值的信息，把这些信息保存下来。

杜韬：SAP作为企业级应用提供商，更关注客户的数据量，而我们的客户有许多数据密集型企业，比如电信、金融、政府、零售等，数据量级从几个TB到数百TB。SAP在德国总部的数据

威廉·张：数据量大并不可怕，问题是要实时处理数据

图注

周俊凌：通过数据访问频率高低确认数据温度，进行数据压缩

中心有3万台服务器，数据量大概是15PB，主要为客户提供服务。我们正在帮助客户将内部应用迁移到我们的数据中心服务平台，这也意味着越来越多的客户数据会存在我们这儿。

面对如此海量的数据，您所在公司是怎样进行存储、处理、分析的？

杜韬：一方面在数据中心，我们使用了标准的虚拟化以及分布式存储；另一方面，我们推出了内存计算技术，用以应对数据应用和分析的挑战。传统的架构存在很大的瓶颈，磁盘读取是以毫秒，而内存读取则是纳秒。因此，我们将以前需要在应用层做的计算分析，比如预测分析或者大量运算，都放到内存里操作，从而实现性能提升，帮助用户充分利用数据。

韩轶平：对Yahoo!的情况，我想分三个部分来说明：数据采集、数据存储和数据处理。

在数据采集方面，我们建立了一个遍布Yahoo!几个数据中心、几十万台机器的实时搜集数据系统，该系统特点是一个主干道负责把数据经过过滤、清理以后，进行整合，并且在高可靠性的情况下，把它放到Hadoop平台。虽然相对来说精度很高、效果很好，但速度会慢一些。为了满足威廉·张所说实时性的需求，还有一个旁路系统，旁路系统在秒级能够把数据汇到主干道上，这是数据采集的部分。

在数据存储方面，基本上以HDFS为核心。

在数据处理方面，主要技术是Hadoop、MapReduce以及我们自己开发的Pig。目前，我们有超过一半数据处理引擎是用Pig完成的。

周俊凌：Teradata一直在持续创新传统的企业级数据仓库产品线，在对接大数据时代的同时，继续传统的BI领域，包括提高数据处理的能力，从而更容易适应大数据管理。例如，通过数据访问频率高低确认数据温度，进行数据压缩，适应大数据的分析要求，使数据管理更容易。

我们有适应超高规模数据容量要求的硬件平台产品Teradata 1000，可以压缩35PB的数据。特别适用一些结构性数据和非结构性数据的分析，同时开发了很多能够进行数据统计和分析的软件包，包括将Hadoop等架构整合到Teradata数据仓库之中，可以基于目前的Teradata企业级数据仓库接口使用。

我们提供基于云的架构，能够使用Amazon EC2，为客户提供安全的存储产品，用来存储公司防火墙以外的、存储在云端的数据。我们刚刚收购了Aster Data公司，它有一些非常好的工具，适用于Hadoop、MapReduce的一些应用。

威廉·张：各互联网企业在云计算技术方面的应用都差不多，比如说百度也用了Hadoop，我提几个比较有特点的地方。

第一个是大搜索，即不仅是把网页抓过来，建立极其庞大的索引，而且为了使数据做到准实时或者更快速的更新，进行一些优化，比如根据地域分布和重要性分布，放在南方或者北方的机房里，主要还是根据数据应用制订的策略。另外就是采用数据流技术。

第二个是机器学习算法。在科技领域里，机器学习以前更多的是对一台服务器内存里的数据进行高复杂的计算，可能要跑很长时间。而在百度，机器学习应用于所有地方，比如判断用户需求，从用户行为反馈中得到我们应该推荐什么样的内容、匹配什么样的广告等，时效性非常高。

可以称得上是增量型、大规模的机器学习方法。

此外，互联网应用要继续发展，最关键还是找到更有价值的数据，即不管数据来自何方，都要按照价值来决定如何处理它。

您怎样看待层出不穷的 oS L技术？

杜韬：我一直认为，存在的就是合理的，NoSQL的产生和演进也是因为我们现有的应用需求所导致。当前在大并发量、海量数据的高效读写等方面，对关系型数据库提出了更高的要求，而NoSQL在这方面有独特的价值和优势。

当然，这并不是说NoSQL的出现就代表着关系型数据库的世界末日，因为对于一些应用，特别是企业级应用，对于事务的一致性以及读写的实时性等各方面有很高的要求，而关系型数据库在这些年的发展中积累了自己的优势。

因此，我很认同NoSQL是“Not Only SQL”的说法，相信在未来关系型数据库和NoSQL会并存甚至是相互融合。

韩轶平：NoSQL是一个很宽泛的概念。在Yahoo!，虽然NoSQL说得不多，但用的NoSQL工具非常多，我们的Key-Value数据库等各种各样的系统，都属于NoSQL框架。至于说NoSQL和SQL之间的关系，因为很多场合需要ACID，也就需要NoSQL的东西，而NoSQL之所以会出现，就像我经常说的“上帝是公平的”，当有一个需求出现时必须放弃另一个东西。我们的很多需求，比如大数据量、高分布性，当有了这些需求以后另一个需求可能成为新的瓶颈。事实上，对我们来说，互联网行业在很多应用中并不需要一致性。当把需求放宽时，自然能够满足另一些需求。

怎样挖掘数据中的价值？

威廉·张：我举一个直观的匹配广告的例子，它包括两类数据：一类是广告库，即广告内容信息和广告客户信息，这类信息很适合于传统数据库；另一类信息是用户看到广告之后的一切行为，经历了日积月累，可能会有几百万亿的用户行为。这两种数据可以相结合，经过机器学习算法就能产生价值。显然，第二种信息更重要，因为它能给用户提供想要的信息，比如搜索一个词，可以利用所有用户在他之前、在他之后的群体智能、群体行为，判定哪一类的信息最重要、最优质，哪一类信息可能是作弊信息，然后经过反馈机制，把最好的内容提供给用户，甚至推荐相关的一些搜索、查询信息。总而言之，对任何企业来说，数据是命根子；对云计算来说，数据处理就是云数据中心或者云计算存在的理由。

韩轶平：我们工作之余经常开玩笑说：从数据中能挖出的东西，不一定是钱，更重要的是用户体验，对互联网公司来说，数据就是一切。Yahoo!不仅仅是搜索引擎，也有很多在美国各领域中排名第一的网站。我们做的很多工作，比如新闻网站信息，都是根据新闻的相关性和大家的兴趣推荐的，我们希望根据每一个用户自己的兴趣，甚至每一个用户此时此刻的兴趣，进行推荐。Yahoo!新闻的推荐系统，是把Yahoo!所有的数据搜集起来，用户在Yahoo!搜索上的所有行为都搜集到一起，做深度挖掘和个性化，对每一个用户都进行分析和推荐，没有这些数据我们不可能为客户提供体验，数据对我们来说就是一切。

韩轶平：Yahoo!不保存原始数据，但做很深入的数据挖掘

图注

杜韬：未来关系型数据库和NoSQL会并存甚至是相互融合

杜韬：既然各位是从互联网的角度来看数据的价值，那么我就从企业的角度来分享一下。

智能电网现在欧洲已经做到了终端，也就是所谓的智能电表。在德国，为了鼓励利用太阳能，会在家庭安装太阳能，除了卖电给你，当你的太阳能有多余电的时候还可以买回来。通过电网收集每隔五分钟或十分钟收集一次数据，收集来的这些数据可以用来预测客户的用电习惯等，从而推断出在未来2~3个月时间里，整个电网大概需要多少电。有了这个预测后，就可以向发电或者供电企业购买一定数量的电。因为电有点像期货一样，如果提前买就会比较便宜，买现货就比较贵。通过这个预测后，可以降低采购成本。

另一个例子更偏我个人的兴趣。丹·布朗的《失落的秘符》一书讲到，如果把很多人的精神集中在一个点，能够移动物体。当然这个我们无从考证，但我们在网上搜索关键词、敏感词时，就可以判断出某件事情的公众态度。有一些新的业务模式，比如做一个网络广告投放评估公司，利用这样的技术评估网络广告的效果，我觉得也许是未来的业务价值产生点。

海量数据时代对企业和技术人员带来了哪些挑战？怎样看待海量数据的未来？

韩轶平：以前我们都说自己是软件工程师，我们这个行业也经常被叫做软件行业，但我认为我们是真正的Information Technology行业。对大多数人来说，现在最重要的一点是转变观念，从Code/Program观念转变成Data观念，在做任何设计和开发时，要把Data放在第一位。

杜韬：海量数据一直在增长，但是我们应该想办法控制下来，未来的趋势应该放在怎样缩小海量数据上，而不是任凭它扩张。此外，海量数据时代对中国来说是一次引领世界IT业的机会。

周俊凌：在云计算时代，业务数据与云紧密结合在一起，提供业务开发的能力，我们从中学到了很多新的东西，有一些东西不再是自己去存储和开发，而是都放在云里面存储。技术产品推向市场的方式与以往相比，发生了很大变化。云的这样一种环境也给数据库提供商带来很多技术上的挑战，例如如何保证存储的安全性，包括身份识别的健全。这关系到数据的存储地方，例如现在发货的数据都是放在全球任何一个地方，不是放在某一个国家里面，这就带来关于数据主权的问题，可能有一些国家和政府不允许把数据放在国家某些地方，这都是一些挑战，需要从技术上解决安全等问题。

威廉·张：这里我浅谈一下两点感受。

首先，数据管理是DBA的一项重要本领，而高校的计算机专业教育里没有特别重视数据程序员，并没有数据管理员；其次，MapReduce并不是一个新概念，早在30~40年前当计算机能力还超小的时候，函数式编程语言就出现了，但至今大学里还没有开设MapReduce或者类似数据处理的课程，也基本上没有人听过这些东西。

未来将所有人的生活经验数据放在云里，这个大概可以实现，但如果解决不好数据安全性问题的话，那么距离最终的实现就会很远。

我期待云计算变成云知识、云智能，而不仅仅是计算的工具。建立数据整合分享是云计算成功的必要和充分条件。

运维技术大势谈

首届中国互联网运维高峰论坛观后感

■ 文 / 杨海朝

网站的稳定运作离不开运维这个幕后群体的“运筹帷幄”，他们是开发人员和机器之间的纽带，只有网站的可用性出现问题时，才会看到他们忙碌的身影。在日前参加的首届中国互联网运维高峰论坛上，我收获良多。

自动化运维

随着网站规模越来越大，机器数量由原来几百台变成了上万台，维护成本越来越高，如何减少成本、减少人为出错的可能性，如何让运维人员不再重复无聊的工作，各大互联网公司都在考虑如何更好地管理机器，使运维工作自动化。

各公司都有自己的Super Agent，采用C/S方式对机器进行批量管理、角色划分，减少管理机器的成本。通过自主研发自动化运维系统或者开源软件Puppet、Anglia、Cfagent、Nagios组合对服务器进行管理。自动化运维系统除了将日常运维中大量重复性工作自动化、集约化、规范化，由过去的手工执行转为自动化操作，从而消除运维工作中的人为失误，实现运维管理的标准化、流程化，提高运维效率、降低运维风险之外，最终上升为为公司提供资产业务分析和产品决策调度。

自动化运维管理系统不是一朝一夕的工作，需要持续的投入，需要通过不断地使用→反馈→完善，最终才能看到需要的结果。

海量数据存储

如何管理好日益膨胀的海量数据，是各大公司都在思考的问题。大家越来越关注如何通过存储结构的更好设计来保证数据的持续存储。

分级存储是在海量数据下必须考虑的问题，数据分为核心数据和非核心数据，不同的数据采用不同的存储策略，分别放在不同的数据库里，其迁移策略和负载均衡策略也完全不一样。定期检查这些数据，将某一段时间不活跃的数据从在线状态迁移到离线状态，在线状态的数据采用高速存储设备，离线状态的数据采用低速设备。

存储应用并不是单纯地一堆存储硬件的堆砌，而是要通过创新的技术和手段，去实现数据的科学管理，目前存储采购的重心在向“硬件＋软件”的整体解决方案迁移，同时在面对海量数据时不能让技术投资比数据的收益增长得更快。

对于存储方案的改造，就像是在重修车轨，不是马上建一套新轨，让一个高速行驶的火车想过来也过不来，而是改造老的车轨，然后不断朝自己的车轨上靠，在遇到瓶颈或性能优化时，刚好可以让它运行在自己预想的车轨上。

云计算就是未来

传统的存储结构因可扩展性问题，当容量和性能的需求增加时，只能通过不断增加高端存储设备，长期下去存储环境会越来越复杂，管理和运营成本会越来越高，同时复杂存储架构和备份方式很难满足日益增长的数据管理和容灾需求。

数据管理技术必须能够高效地管理大数据集，同时能在规模庞大的数据中找到需要的数据。从海量数据中获得有价值的信息，为用户提供良好的用户体验，增强企业的竞争力，这对企业来说是一个机遇也是一个挑战。云计算系统中的数据管理技术有两种：Google的BigTable数据管理技术和Hadoop的数据管理模块HBase。

无论个人还是企业无须掌握任何IT技术，只需要利用终端设备，借助各种灵活的网络接入手段，通过“云”就可以完成所需要的任何计算和存储，获得无穷无尽的计算和存储资源，最终的结果是云计算虚拟化让实时海量数据处理无忧。期待云技术完全普及的那一天早日到来。

NoSQL生态系统

■ 文 / Adam Marcus 译 / iammutex

何为NoSQL？NoSQL不是一个工具，而是由一些具有互补性和竞争性的工具组成的一个概念，是一个生态圈。这些被称为NoSQL的工具，在存储数据的方式上，提供了一种与（基于SQL语言的）关系型数据库截然不同的思路。要想了解NoSQL，必须先了解现有的这些工具，去理解那些引导它们开拓出新的存储领域的设计思路。

NoSQL其名

在给NoSQL下定义之前，我们先来试着从它的名字上做一下解读。顾名思义，NoSQL系统的数据操作接口应该是非SQL类型的。但在NoSQL社区，NoSQL被赋予了更具有包容性的含义，其意为Not Only SQL，即NoSQL提供了一种与传统关系型数据库不同的存储模式，这为开发者提供了关系型数据库之外的另一种选择。

NoSQL的启示

NoSQL运动受到了很多相关研究论文的启示，在所有资料中，最核心的有两个：Google的BigTable论文和Amazon的Dynamo论文。

特性概述

NoSQL系统舍弃了一些SQL标准中的功能，取而代之的是一些简单灵活的功能。NoSQL的构建思想就是尽量简化数据操作，尽量让操作的执行效率可预估。当你去考查一个NoSQL系统时，下面的几点是值得注意的。

- **数据模型及操作模型**：你的应用层数据模型是行、对象还是文档型的呢？这个系统是否能支持你进行一些统计工作呢？
- **可靠性**：当你更新数据时，新的数据是否立刻写到持久化存储中去了？新的数据是否同步到多台机器上了？
- **扩展性**：你的数据量有多大，单机是否能容下？你的读写量需求单机是否能支持？
- **分区策略**：考虑到对扩展性、可用性或者持久性的要求，你是否需要一份数据被存在多台机器上？你是否需要知道或者说你能否知道数据在哪台机器上？
- **一致性**：你的数据是否被复制到了多台机器上？这些不同节点的数据如何保证一致性？
- **事务机制**：业务是否需要ACID事务机制？
- **单机性能**：如果你打算持久化的将数据存在磁盘上，哪种数据结构能满足你的需求（你的需求是读多还是写多）？写操作是否会成为磁盘瓶颈？
- **负载可评估**：对于一个读多写少的应用，诸如响应用户请求的网络应用，我们总会花很多精力来关注负载情况。你可能需要进行数据规模的监控，对多个用户的数据进行汇总统计。你的应用场景是否需要这样的功能呢？

NoSQL数据模型及操作模型

数据库的数据模型指的是数据在数据库中的组织方式，数据库的操作模型指的是存取这些数据的方式。通常数据模型包括关系模型、键值模型以及各种图结构模型。操作语言可能包括SQL、键值查询及MapReduce等。NoSQL通常结合了多种数据模型和操作模型，提供不一样的架构方式。

基于Key值存储的NoSQL数据模型

在键值型系统中，复杂的联合查询以及满足多个条件的数据查询操作就不那么容易实现了，需要换一种思维来建立和使用键名。比如要获取部门号为20的所有员工的信息，应用层可以先获取Key为employee_departments:20的这个列表，然后再循环地拿这个列表中的ID通过获取employee:ID得到所有员工的信息。

- 存储

Key-Value存储可以说是最简单的NoSQL存储，每个Key值对应一个任意的数据值。对NoSQL系统来说，这个任意的数据值是什么，它并不关心。比如在员工信念数据库里，employee:30这个Key对应的可能就是一段包含员工所有信息的二进制数据。这个二进制的格式可能是Protocol Buffer、Thrift或者Avro都无所谓。

- 结构化数据存储

Key-结构化数据存储的典型代表是Redis，Redis将Key-Value存储的Value变成了结构化的数据类型。Value的类型包括数字、字符串、列表、集合以及有序集合。除了set/get/delete操作以为，Redis还提供了很多针对以上数据类型的特殊操作，比如针对数字可以执行增、减操作，对list可以执行push/pop操作，通过提供这种针对单个Value进行的特定类型的操作，Redis可以说实现了功能与性能的平衡。

- 文档存储

Key-文档存储的代表有CouchDB、MongoDB和Riak。这种存储结构下Key-Value的Value是结构化的文档，通常这些文档是被转换成JSON或者类似于JSON的结构进行存储。文档可以存储列表，键值对以及层次结构复杂的文档。

- i 的列簇式存储

HBase和Cassandra的数据模型都借鉴自Google的BigTable。这种数据模型的特点是列式存储，每一行数据的各项被存储在不同的列中（这些列的集合称作列簇）。而每一列中每一个数据都包含一个时间戳属性，这样列中的同一个数据项的多个版本都能保存下来。

列式存储可以这样理解：将行ID、列簇号，列号以及时间戳一起，组成一个Key，然后将Value按Key的顺序进行存储。Key值的结构化使这种数据结构能够实现一些特别的功能，最常用的就是将一个数据的多个版本存成时间戳不同的几个值，这样就能方便地保存历史数据。这种结构也能天然地进行高效的松散列数据（在很多行中并没有某列的数据）存储。当然，对于那些很少有某一行有NULL值的列，由于每一个数据必须包含列标识，这又会造成空间的浪费。

图结构存储

图结构存储是NoSQL的另一种存储实现。其指导思想是：数据并非对等的，关系型的存储或者键值对的存储，可能都不是最好的存储方式。图结构是计算机科学的基础结构之一，Neo4j和HyperGraphDB是当前最流行的图结构数据库。

复杂查询

在NoSQL存储系统中，有很多比键值查找更复杂的操作。比如MongoDB可以在任意数据行上建立索引，可以使用Javascript语法设定复杂的查询条件。BigTable型的系统通常支持对单独某一行的数据进行遍历，允许对单列的数据进行按特定条件的筛选。CouchDB允许你创建同一份数据的多个视图，通过运行MapReduce任务来实现一些更为复杂的查询或者更新操作。很多NoSQL系统都支持与Hadoop或者其他MapReduce框架结合来进行一些大规模数据分析工作。

事务机制

与关系型数据库不同的是，NoSQL系统通常注重性能和扩展性，而非事务机制。传统的SQL数据库的事务通常都是支持ACID的强事务机制。ACID的支持使得应用者能够很清楚他们当前的数据状态。对很多NoSQL系统来说，对性能的考虑远在ACID的保证之上。通常NoSQL系统仅提供行级别的原子性保证，也就是说同时对同一个Key下的数据进行的两个操作，在实际执行时是会串行的，保证了每一个Key-Value对不会被破坏。

Schema-free的存储

还有一个很多NoSQL的共同点，就是它通常并没有强制的数据结构约束。即使是在文档型存储或者列式存储上，也不会要求某一个数据列在每一行数据上都必须存在。

数据可靠性

最理想的状态是，数据库会把所有写操作立刻写到持久化存储的设备，同时复制多个副本到不同地理位置的不同节点上，以防止数据丢失。但这种对数据安全性的要求对性能是有影响的，所以不同的NoSQL系统在自身性能的考虑下，在数据安全上采取了不太一样的策略。

单机可靠性

单机可靠性理解起来非常简单，它的定义是写操作不会由于机器重启或者断电而丢失。通常单机可靠性的保证是通过把数据写到磁盘来完成的，而这通常会造成磁盘I/O成为整个系统的瓶颈。下面我们谈谈一些在单机可靠性的保证下提高性能的方法。

- **控制fsync的调用频率**

Redis提供了几种对fsync调用频率的控制方法。应用开发者可以配置Redis在每次更新操作后都执行一次fsync，这样会比较安全，当然也就比较慢。Redis也可以设置成N秒种调用一次fsync，这样性能会更好一点。但这样的后果就是一旦出现故障，最多可能导致N秒内的数据丢失。而对一些可靠性要求不太高的场合（比如仅仅把Redis当Cache用的时候），应用开发者甚至可以直接关掉fsync的调用：让操作系统来决定什么时候需要把数据flush到磁盘（译者注：这只是Redis append only file的机制，Redis是可以关闭aof日志的，另外，Redis本身支持将内存中数据dump成rdb文件的机制，和上面说的不是一回事）。

- **使用日志型的数据结构**

Cassandra、HBase、Redis和Riak都会把写操作顺序的写入到一个日志文件中。相对于存储系统中的其他数据结构，上面说到的日志文件可以频繁地进行fsync操作，这样就把对磁盘的随机写变成顺序写了。

- **通过合并写操作提高吞吐性能**

Cassandra有一个机制，它会把一小段时间内的几个并发的写操作放在一起进行一次fsync调用，这种做法叫group commit。

多机可靠性

由于硬件层面有时会造成无法恢复的损坏，单机可靠性的保证在这时就鞭长莫及了。对于一些重要数据，跨机器做备份保存是必备的安全措施。一些NoSQL系统提供了多机可靠性的支持。

- Redis采用传统的主从数据同步的方式。
- MongoDB提供了一种叫Replica Sets高可用架构。
- Riak、Cassandra和Voldemort提供了一些更灵活的可配置策略，并提供一个可配置的参数N，代表每一个数据会被备份的份数。为了应对整个数据中心出现故障的情况，需要实现跨数据中心的多机备份功能。

横向扩展带来性能提升

横向扩展的目标是达到线性的效果，即如果你增加一倍的机器，那么负载能力应该也能相应的增加一倍。其主要需要解决的问题是如何让数据在多台机器间分布，这里面涉及到分片技术。

分片的意思，就是没有任何一台机器可以处理所有写请求，也没有任何一台机器可以处理对所有数据的读请求。下面我们将会对hash分片和范围分片两种分片方式进行描述。

如非必要，请勿分片

分片会导致系统复杂程度大增，所以，如果没有必要，请不要使用分片。普通情况下，我们可以使用读写分离和构建缓存的方式来缓解我们的数据读压力。但如果写操作达到单点无法承担的程度，那我们可能就真的需要进行分片了。

通过协调器进行数据分片

一种分片策略是通过引入一个中间代理层来实现，该代理层记录数据在各个节点的分布状况，所有读写请求都通过代理层来做路由。比如与CouchDB的两个项目：Lounge和BigCouch。类似的，Twitter自己也实现了一个叫Gizzard的协调器，可以实现数据分片和备份功能。

一致性hash环算法

一致性hash是一种被广泛应用的技术，其最早在一个叫distributed hash tables（DHTs）的系统中进行使用。那些类Dynamo的应用，比如Cassandra、Voldemort和Riak，基本上都使用了一

致性hash环算法。

如图1所示，一致性hash环算法有一个hash函数H，所有存储数据的节点和数据本身都可以通过这个函数算出一个hash值，作为自己在下面环上的位置。然后每个节点会负责存储其hash值到下一个节点间的所有数据的存储。这样使得即使节点数变化了，大部分数据并不需要进行迁移。

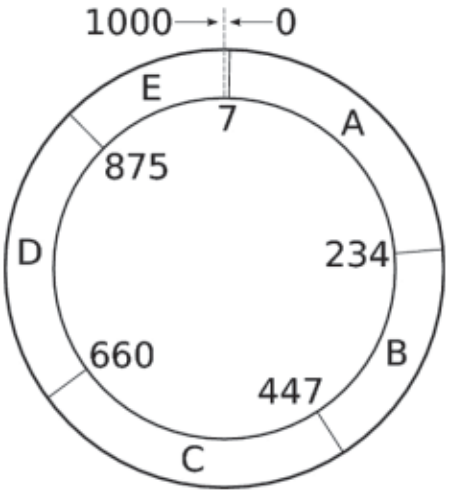


图1　一致性hash环算法的hash函数

连续范围分区

使用连续范围分区的方法进行数据分片，需要我们保存一份映射关系表，标明哪一段Key值对应存在哪台机器上。与一致性hash类似，连续范围分区会把Key值按连续的范围分段，每段数据会被指定保存在某个节点上，然后会被冗余备份到其他节点。

- i　　的处理方式

Google BigTable论文中描述了一种范围分区方式，它将数据切分成一个个的tablet数据块。每个tablet保存一定数量的键值对。然后存储在Tablet 服务器上。tablet块的大小会保持在一定范围，太大的块会分裂成两个，太小的块又会合并成一个。BigTable通过一个叫Chubby的模块来实现节点状态检测。类似的在Hadoop中有一个叫ZooKeeper的工具实现此功能。

一致性

上面讲到了通过将数据冗余存储到不同的节点来保证数据安全和减轻负载，下面我们来看看这样做引发的一个问题：保证数据在多个节点间的一致性是非常困难的。在多个点间保持数据的一致性的问题，也就是本章的主题。下面我们首先来看一下在著名的CAP理论。

- 一致性（C）：在分布式系统中的所有数据备份，在同一时刻是否同样的值。
- 可用性（A）：在集群中一部分节点故障后，集群整体是否还能响应客户端的读写请求。
- 分区容忍性（P）：集群中的某些节点在无法联系后，集群整体是否还能继续进行服务。

而CAP理论就是说在分布式存储系统中，最多只能实现上面的两点。再加之当前的网络硬件肯定会出现延迟丢包等问题，所以分区容忍性是我们必须需要实现的。结果就是我们只能在一致性和可用性之间进行权衡，没有NoSQL系统能同时保证这三点。

对一致性的保证，通常有强一致性和弱一致性的选择，而在弱一致性里，又以最终一致性的实现较为普遍。

如果我们采用NRW的设定，N为数据需要备份的份数，R为读操作需要读到的不同节点上的数据份数，W为写操作需要成功写到不同节点的数据份数，那么当R＋W>N时，既是强一致性的保证，当R+W<N时，就是弱一致性。在弱一致性中，可以通过vector clock多版本控制等方法，来实现数据的最终一致性。

写在最后的话

目前NoSQL系统来处在它的萌芽期，我们上面讨论到的很多NoSQL系统，它们的架构、设计和接口可能都会改变。本章的目的，不在于让你了解这些NoSQL系统目前是如何工作的，而在于让你理解这些系统之所以这样实现的原因。NoSQL系统把更多的设计工作留给了应用开发工作者来做。理解上面这些组件的架构，不仅能让你写出下一个NoSQL系统，更让你对现有系统应用得更好。

（编者注：本文根据NoSQLFan网站原载同名文章http://blog.nosqlfan.com/html/2 7 .html整理而成，英文原文链接为http://www.aosabook.org/en/nosql.html）

E:提升资源利用率的 d 框架

■ 文 / 杨栋

Hadoop系统提供了MapReduce计算框架的开源实现，像Yahoo!、Facebook、淘宝、中移动、百度、腾讯等公司都在借助Hadoop进行海量数据处理。Hadoop系统性能不仅取决于任务调度器的分配策略，还受到分配后实际任务执行效率的影响，任务执行常常涉及读取、排序、归并、压缩、写入等具体阶段。

HCE计算框架是一个开源项目，旨在通过优化任务执行的各个阶段，提升整个Hadoop系统的效率。与Hadoop Java框架相比，基于HCE框架的MapReduce任务最高可以节省超过30%的CPU资源使用。

图1给出了HCE框架在Hadoop生态系统中所处的位置。对于OLTP系统来说，用户通过Web前端生成相应请求，请求经过中间件处理，作为数据进入数据库或者K-V存储系统中，同时会产生日志。OLTP系统产生的数据和日志都会作为分析系统的输入，对于搜索引擎和广告系统来说，每天的日志会轻松超过TB。日志和业务数据一般会存放到海量存储系统HDFS文件系统或者K-V存储系统中，分布式计算框架MapReduce一般会基于存储系统之上。每天会执行成千上万的MapReduce作业进行海量数据处理，产生的结果会有三个去处：存放于海量存储系统以备后续使用；导入用于产生报表或分析的数据库；作为OLTP系统的输入，导入线上存储中。MapReduce作业一般由内部用户通过Hadoop原生客户端、Pig/DISQL语言客户端或者Hive数据仓库三种方式进行提交，作业执行结果可以通过SQL客户端查询。

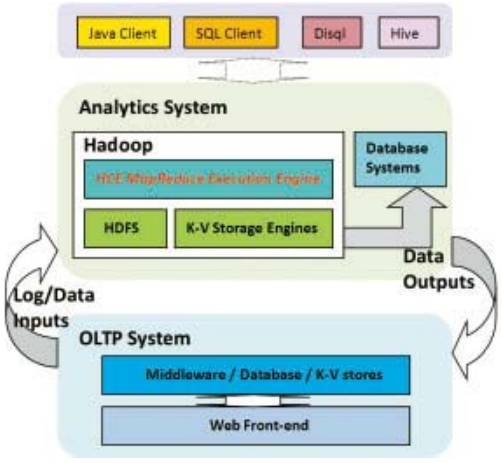


图1 Hadoop生态系统中的HCE计算框架

问题

越来越多的公司开始使用Hadoop及其周边系统进行海量数据分析，Yahoo!和Facebook的Hadoop集群节点数已经过万，并且节点增长的趋势不减，国内公司如腾讯和百度等也面临着同样的问题。不断增长的业务需求和业务数据导致的集群资源紧缺是集群不断扩容的主要原因，CPU资源（注：存储资源紧缺的解决方案之一是开启重量级的压缩（如Facebook），这涉及CPU资源的使用）又是其中最为紧缺的。为了控制成本，优化集群资源利用率势在必行。

对于分布式计算层面，资源优化有两个途径：一是通过精细化的资源调度来保证全局资源的最大化利用，这通常涉及合理的资源调度算法和轻量级的资源隔离；二是通过优化计算任务和用户程序来提升原有计算作业的资源使用率。HCE计算框架主要关注后者。

跨平台、高扩展、通用接口的计算框架也带来了额外开销

分析Hadoop MapReduce计算框架，已有实现可以再做权衡。

MapReduce框架通过Java语言高效实现，保证了其跨平台的兼容性；不过国内互联网公司一般都使用Intel x86平台，兼容性的优势很难得以体现，因此可以选择优于Java性能但不支持跨平台的语言来实现MapReduce框架。

为了最大化可扩展性（Extensibility），Hadoop实现了多层级的数据处理流封装，这使得Java框架在大数据处理时存在一些性能损耗，实际上可以实现更加直接的数据处理路径来提升处理效率。

国内互联网公司的工程师们大多使用C++进行功能开发或脚本语言来处理文本，Java接口对于他们而言是比较麻烦的事情。Hadoop提供了Streaming编程接口，允许用户程序可以通过媒介——标准输入输出来和计算框架交互数据，也即绕开了语言的限制，因此很多用户任务是通过Streaming接口启动的。Streaming接口的优势在于支持多语言开发，而增加通用性带来的是性能的损耗，即数据拷贝管道和Key切分开销（大约2%～5%），并且不如原生态的语言接口更加适宜编程。

用户程序在计算框架控制之外

除了框架的开销，计算任务的资源占用还包括用户程序。如图2所示，Hadoop Streaming和Pipes框架支持C++用户开发MapReduce应用程序，框架启动用户可执行程序，框架和用户程序分别处于两个进程，分别占用资源。简单的分析程序不会占用太多的CPU资源，即用户程序在整个计算任务执行时间中所占比例不大，此时，优化计算框架会带来比较可观的收益；不过，对于复杂的分析程序而言，用户程序所占时间远远超过计算框架，此时，优化计算框架带来的收益可能微乎其微。因此，节省集群CPU资源离不开优化用户程序。

优化用户作业可以分为两个层面，一是通过较高层次的统一入口让用户提交作业，例如使用数据仓库Hive的用户不会操作Hadoop MapReduce的API，在Hive内部统一做优化，包括一些静态或者动态方法，调整用户作业参数，使任务利用最少资源高效执行；二是优化直接操作MapReduce API的用户作业，当然Hive也属于这

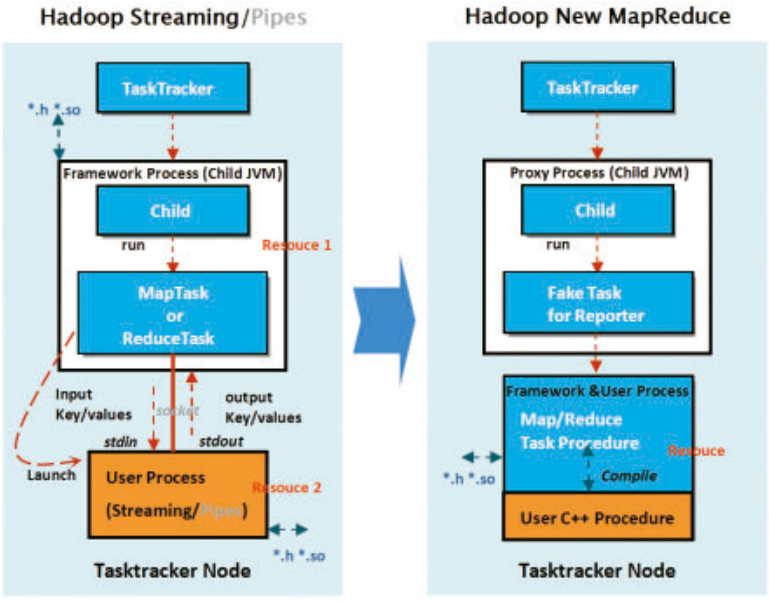


图2 MapReduce任务执行框架示意图

个范畴。设想用户作业或者数据仓库是通过C++语言实现的，编译由用户实施，平台仅仅通过接口调用用户的可执行程序，那么此时想要优化用户程序会比较困难。有动态和静态两种优化方式：动态优化方式（注：详情参见*Starfish: A Selftuning System for Big Data Analytics (CIDR'11)*）是在MapReduce上新增一层，通过profiler和sampler等技术动态调整作业参数；静态优化方式是让用户用编译时依赖框架提供的头文件和库文件，通过编译优化技术提升用户程序性能。

解决

HCE计算框架通过静态优化计算框架和用户程序来提升计算任务的CPU使用率。如图2所示，将框架和用户程序整合到一个进程，可以通过同一套编译机制同时优化框架和用户程序，Key-Values处理也处于同一个进程空间，不用借助媒介（管道或套接字）来传递。HCE和Hadoop提供的用户编程接口如表1所示。

计算框架高效C++实现

HCE框架通过C++语言实现了MapReduce的数据处理逻辑，依托比Java性能更优的C++语言，可以在数据处理操作上获得更佳的CPU利用率，同时也可以更加直接地调用Native Lib而非通过JNI

表1 MapReduce框架各用户接口对比

Interface	Resource Utility	Performance	Practicability	Universality
Java (Hadoop)	基准	基准	基于Java开发	Facebook(Hive)大量使用
Streaming (Hadoop)	用户程序自行优化	低于Java 5%	读写标准输入/输出	国内公司大量使用
C++ (HCE)	框架优化；用户优化节省CPU最多	提升5%~30%，比较Java	基于C++库开发	数据仓库和重负载作业使用
Streaming (HCE)	框架优化 框架节省CPU	提升10%以上，比较Streaming(Java)	读写标准输入/输出	透明替换Streaming(Java)
Python (HCE)	框架优化 框架节省CPU	提升15%以上，比较Streaming(Java)	基于Python库开发	脚本开发者使用

（注：压缩库是Native实现，Hadoop通过JNI来调用压缩方法，HCE压缩在一个进程空间执行）；此外，通过高效的编译优化方法，例如ICC编译器等，可以进一步挖掘框架的性能优势。

HCE框架通过精简的方式实现了MapReduce的数据处理流程，比较多层次的Java流式封装，HCE的处理流程更加高效。

HCE框架提供了多种语言接口C++、Python等，方便了用户编程，也节省了Streaming接口的额外开销；同时HCE也提供完全兼容原有Java Streaming的接口，即原有作业可以无缝迁移到HCE框架。

用户程序静态编译优化

HCE框架采用的是静态优化用户程序的方式，动态优化交给上层的数据仓库去做。对于那些CPU负载较重的用户程序，HCE提供C++编程接口给用户，用户编译本地程序需要依赖框架的头文件和库文件，头文件中内置了如SSE等优化代码，可以使得用户程序在编译时被优化。这种简单的方式能够使得用户程序的执行效率大幅提升。

框架

HCE框架实现了Hadoop支持的功能组件，例如在C++空间中支持Text或者SequenceFile格式的RecordReader和RecordWriter，也支持Gzip、Lzo、QuickLz、Lzma等4种压缩格式。由于输入文件切分是在Hadoop Client实施的，所以Split方法还是在Java空间执行的；当然，用户定义的Mapper和Reducer必须在C++空间实现，例如Hive想要基于HCE框架执行，那么就必须实现C++版本的Mapper、Reducer等功能组件。

图3展示了HCE框架的数据处理流程，可以看出HCE框架在C++空间高效实现了多个可扩展的功能模块，如RecordReader、OutputCollector、Shuffle、ReduceInputReader、RecordWriter、Committer、Partitioner、Mapper、Reducer、Combiner等，处理逻辑比Hadoop MapReduce更加紧凑高效。处于Hadoop Java空间的MapRunner和ReduceRunner只是起到收集状态信息的作用。

HCE框架的性能提升主要集中在Map阶段，大约超过40%。对于一般的MapReduce程序，相比

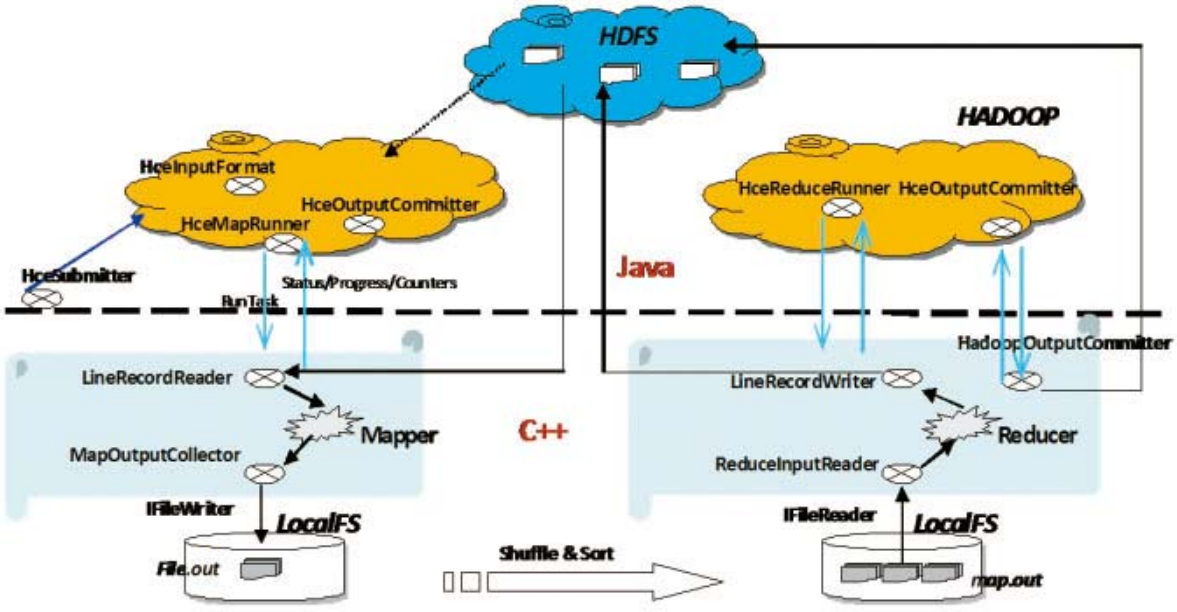


图3 HCE框架数据处理流程图

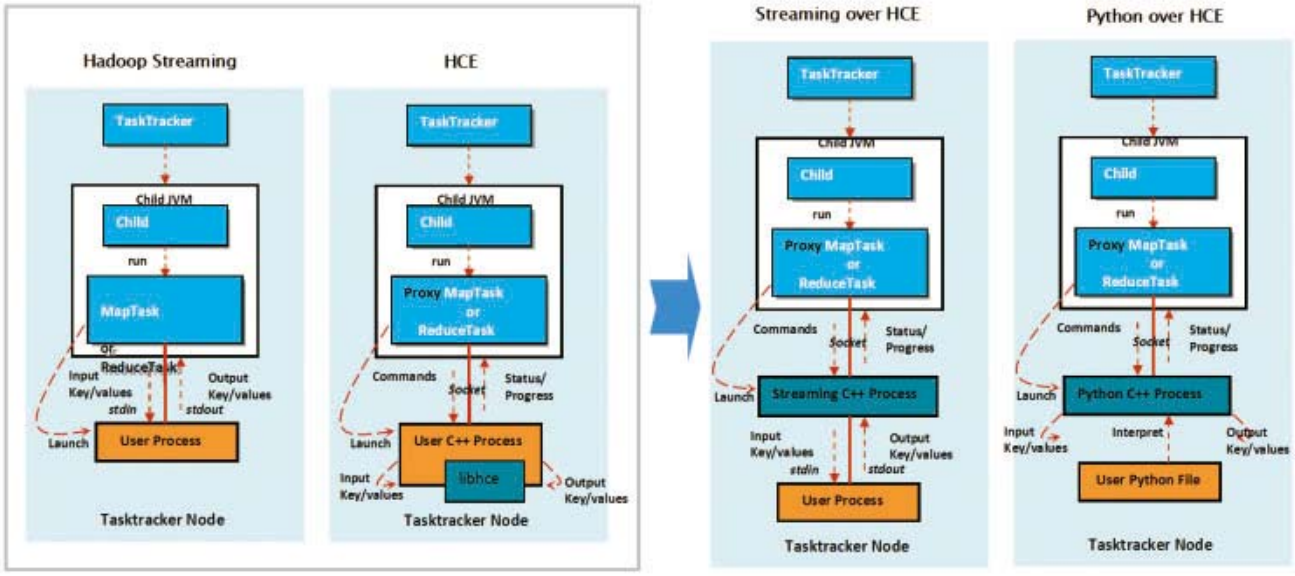


图4　Streaming Over HCE和Python Over HCE框架示意图

Shuffle和Reduce阶段，Map阶段也是其资源占用最多的阶段，因为最终作业的输出一般仅仅是输入的10%，大量的数据处理是在Map阶段完成的。

拓展

仅有基本的HCE框架是不够的，因为大量已有作业是通过Streaming接口执行的，而且除C++开发接口之外，脚本开发者也想使用对应语言的开发接口。幸运的是，所有脚本语言都基于C开发，因此可以实现简单的解释器，将脚本语言翻译成C语言，最终执行的仍是HCE框架，而这个解释开销很小。当然，Streaming作业的额外开销是不可避免了，不过基于HCE框架的Streaming作业可以利用框架的性能优势获得CPU利用率的提升，这对于轻量级作业收益仍然可观。

图4展示了Java Streaming、HCE、Streaming Over HCE和Python Over HCE四个框架的数据处理途径。Java Streaming框架的数据处理仍然是在Java空间完成的，而HCE、Streaming Over HCE、Python Over HCE框架的数据处理都在C++空间完成，Child JVM仅从HCE进程收集任务状态信息。

未来

MapReduce计算框架并不仅仅依赖于HDFS存储系统，也可以基于其他存储系统，例如Hypertable或者是其他的K-V系统。目前很多块存储系统或K-V系统都是C++语言实现的，想要在其上使用原生态的Hadoop MapReduce，就必须通过存储系统的语言转换接口（例如Hypertable的Thrift）或者计算系统的转换接口（例如Hadoop的AvroRPC等），问题是数据的序列化和反序列化难免带来额外开销。因此，基于HCE框架、非Java语言实现的存储系统可以更加高效地支持Hadoop MapReduce计算，当然它们需要实现对应的Split、RecordReader、RecordWriter和Committer等组件。

小结

HCE框架是Hadoop MapReduce框架的一个衍生品。依托HCE框架的高效本地处理机制，Hadoop作业可以最多节省30%的CPU资源使用。此外，HCE提供了C++、Python等多种编程接口，并且保证了已有接口的向前兼容；多种编译优化技术也可以方便地应用到MapReduce框架；最后，HCE通过编译优化和内置解释器等方式优化了用户程序的执行。

杨栋

百度分布式高级研发工程师，从事Hypertable、Hadoop及流式计算的研究和开发。

淘宝数据魔方技术架构解析

■ 文 / 张轩丞

淘宝网拥有国内最具商业价值的海量数据。截至当前，每天有超过30亿的店铺、商品浏览记录，10亿在线商品数，上千万的成交、收藏和评价数据。如何从这些数据中挖掘出真正的商业价值，进而帮助淘宝、商家进行企业的数据化运营，帮助消费者进行理性的购物决策，是淘宝数据平台与产品部的使命。

为此，我们进行了一系列数据产品的研发，比如为大家所熟知的量子统计、数据魔方和淘宝指数等。尽管从业务层面来讲，数据产品的研发难度并不高；但在“海量”的限定下，数据产品的计算、存储和检索难度陡然上升。本文将以数据魔方为例，向大家介绍淘宝在海量数据产品技术架构方面的探索。

淘宝海量数据产品技术架构

数据产品的一个最大特点是数据的非实时写入，正因为如此，我们可以认为，在一定的时间段内，整个系统的数据是只读的。这为我们设计缓存奠定了非常重要的基础。

图1 淘宝海量数据产品技术架构

按照数据的流向来划分，我们把淘宝数据产品的技术架构分为五层（如图1所示），分别是数据源、计算层、存储层、查询层和产品层。位于架构顶端的是我们的数据来源层，这里有淘宝主站的用户、店铺、商品和交易等数据库，还有用户的浏览、搜索等行为日志等。这一系列的数据是数据产品最原始的生命力所在。

在数据源层实时产生的数据，通过淘宝自主研发的数据传输组件DataX、DbSync和Timetunnel准实时地传输到一个有1500个节点的Hadoop集群上，这个集群我们称之为“云梯”，是计算层的主要组成部分。在“云梯”上，我们每天有大约40000个作业对1.5PB的原始数据按照产品需求进行不同的MapReduce计算。这一计算过程通常都能在凌晨两点之前完成。相对于前端产品看到的数据，这里的计算结果很可能是一个处于中间状态的结果，这往往是在数据冗余与前端计算之间做了适当平衡的结果。

不得不提的是，一些对实效性要求很高的数据，例如针对搜索词的统计数据，我们希望能尽快推送到数据产品前端。这种需求再采用“云梯”来计算效率将是比较低的，为此我们做了流式数据的实时计算平台，称之为“银河”。“银河”也是一个分布式系统，它接收来自TimeTunnel的实时消息，在内存中做实时计算，并把计算结果在尽可能短的时间内刷新到NoSQL存储设备中，供前端产品调用。

容易理解，“云梯”或者“银河”并不适

合直接向产品提供实时的数据查询服务。这是因为，对于“云梯”来说，它的定位只是做离线计算的，无法支持较高的性能和并发需求；而对于“银河”而言，尽管所有的代码都掌握在我们手中，但要完整地将数据接收、实时计算、存储和查询等功能集成在一个分布式系统中，避免不了分层，最终仍然落到了目前的架构上。

为此，我们针对前端产品设计了专门的存储层。在这一层，我们有基于MySQL的分布式关系型数据库集群MyFOX和基于HBase的NoSQL存储集群Prom，在后面的文字中，我将重点介绍这两个集群的实现原理。除此之外，其他第三方的模块也被我们纳入存储层的范畴。

存储层异构模块的增多，对前端产品的使用带来了挑战。为此，我们设计了通用的数据中间层——glider——来屏蔽这个影响。glider以HTTP协议对外提供restful方式的接口。数据产品可以通过一个唯一的URL获取到它想要的数据。

以上是淘宝海量数据产品在技术架构方面的一个概括性的介绍，接下来我将重点从四个方面阐述数据魔方设计上的特点。

关系型数据库仍然是王道

关系型数据库（RDBMS）自20世纪70年代提出以来，在工业生产中得到了广泛的使用。经过三十多年的长足发展，诞生了一批优秀的数据库软件，例如Oracle、MySQL、DB2、Sybase和SQL Server等。

尽管相对于非关系型数据库而言，关系型数据库在分区容忍性（Tolerance to Network Partitions）方面存在劣势，但由于它强大的语义表达能力以及数据之间的关系表达能力，在数据产品中仍然占据着不可替代的作用。

淘宝数据产品选择MySQL的MyISAM引擎作为底层的数据存储引擎。在此基础上，为了应对海量数据，我们设计了分布式MySQL集群的查询代理层——MyFOX，使得分区对前端应用透明。

目前，存储在MyFOX中的统计结果数据已

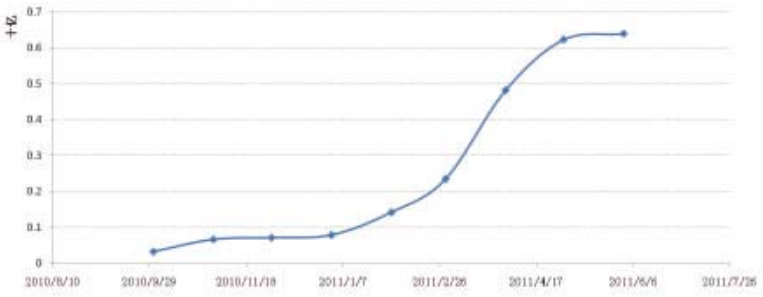


图2　MyFOX中的数据增长曲线

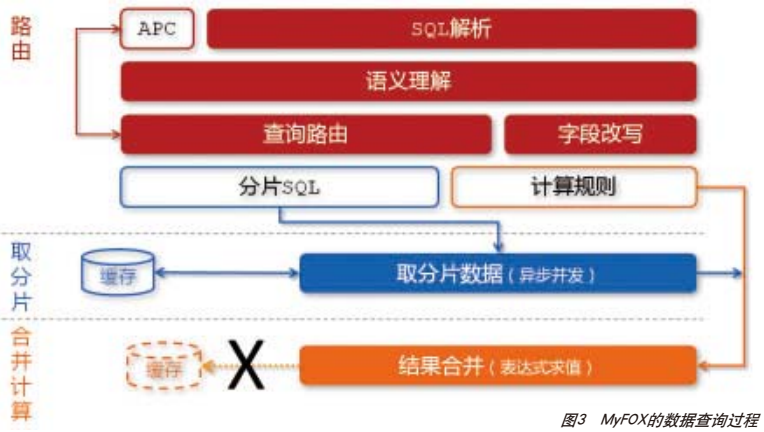


图3　MyFOX的数据查询过程

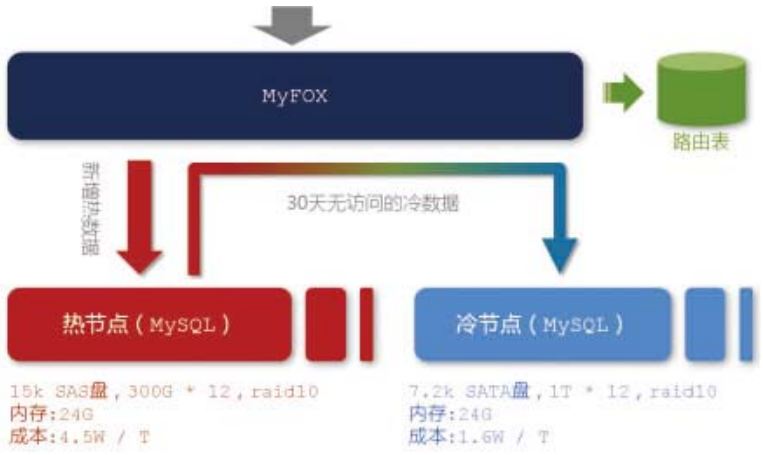


图4　MyFOX节点结构

经达到10TB，占据着数据魔方总数据量的95%以上，并且正在以每天超过6亿的增量增长着（如图2所示）。这些数据被我们近似均匀地分布到20个MySQL节点上，在查询时，经由MyFOX透明地对外服务（如图3所示）。

值得一提的是，在MyFOX现有的20个节点中，并不是所有节点都是“平等”的。一般而

言，数据产品的用户更多地只关心“最近几天”的数据，越早的数据，越容易被冷落。为此，出于硬件成本考虑，我们在这20个节点中分出了“热节点”和“冷节点”（如图4所示）。

顾名思义，“热节点”存放最新的、被访问频率较高的数据。对于这部分数据，我们希望能给用户提供尽可能快的查询速度，所以在硬盘方面，我们选择了每分钟15000转的SAS硬盘，按照一个节点两台机器来计算，单位数据的存储成本约为4.5W/TB。相对应地，“冷数据”我们选择了每分钟7500转的SATA硬盘，单碟上能够存放更多的数据，存储成本约为1.6W/TB。

将冷热数据进行分离的另外一个好处是可以有效降低内存磁盘比。从图4可以看出，“热节点”上单机只有24GB内存，而磁盘装满大约有1.8TB（300 * 12 * 0.5 / 1024），内存磁盘比约为4:300，远远低于MySQL服务器的一个合理值。内存磁盘比过低导致的后果是，总有一天，即使所有内存用完也存不下数据的索引了——这个时候，大量的查询请求都需要从磁盘中读取索引，效率大打折扣。

图5 全属性选择器

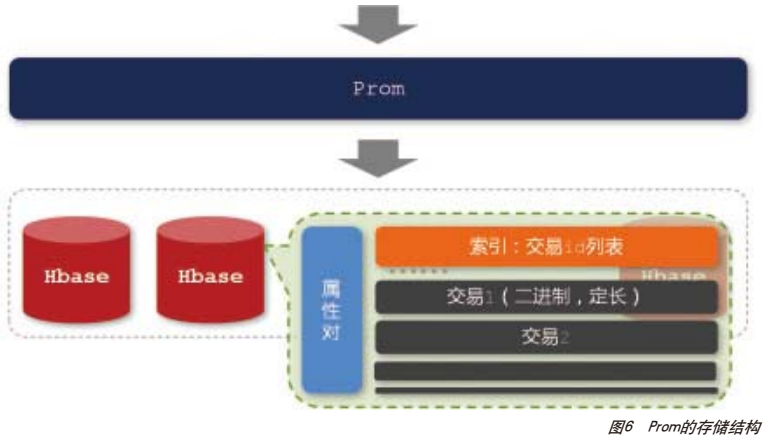


图6 Prom的存储结构

NoSQL是SQL的有益补充

在MyFOX出现之后，一切都看起来那么完美，开发人员甚至不会意识到MyFOX的存在，一条不用任何特殊修饰的SQL语句就可以满足需求。这个状态持续了很长一段时间，直到有一天，我们碰到了传统的关系型数据库无法解决的问题——全属性选择器（如图5所示）。

这是一个非常典型的例子。为了说明问题，我们仍然以关系型数据库的思路来描述。对于笔记本电脑这个类目，用户某一次查询所选择的过滤条件可能包括“笔记本尺寸”、“笔记本定位”、“硬盘容量”等一系列属性（字段），并且在每个可能用在过滤条件的属性上，属性值的分布是极不均匀的。在图5中我们可以看到，笔记本电脑的尺寸这一属性有着10个枚举值，而“蓝牙功能”这个属性值是个布尔值，数据的筛选性非常差。

在用户所选择的过滤条件不确定的情况下，解决全属性问题的思路有两个：一个是穷举所有可能的过滤条件组合，在“云梯”上进行预先计算，存入数据库供查询；另一个是存储原始数据，在用户查询时根据过滤条件筛选出相应的记录进行现场计算。很明显，由于过滤条件的排列组合几乎是无法穷举的，第一种方案在现实中是不可取的；而第二种方案中，原始数据存储在什么地方？如果仍然用关系型数据库，那么你打算怎样为这个表建立索引？

这一系列问题把我们引到了“创建定制化的存储、现场计算并提供查询服务的引擎”的思路上来，这就是Prometheus（如图6所示）。

从图6可以看出，我们选择了HBase作为Prom的底层存储引擎。之所以选择HBase，主要是因为它是建立在HDFS之上的，并且对于MapReduce有良好的编程接口。尽管Prom是一个通用的、解决共性问题的服务框架，但在这里，我们仍然以全属性选择为例，来说明Prom的工作原理。这里的原始数据是前一天在淘宝上的交易明细，在HBase集群中，我们以属性

对（属性与属性值的组合）作为row-key进行存储。而row-key对应的值，我们设计了两个column-family，即存放交易ID列表的index字段和原始交易明细的data字段。在存储的时候，我们有意识地让每个字段中的每一个元素都是定长的，这是为了支持通过偏移量快速地找到相应记录，避免复杂的查找算法和磁盘的大量随机读取请求。

图7用一个典型的例子描述的Prom在提供查询服务时的工作原理，限于篇幅，这里不做详细描述。值得一提的是，Prom支持的计算并不仅限于求和SUM运算，统计意义上的常用计算都是支持的。在现场计算方面，我们对Hbase进行了扩展，Prom要求每个节点返回的数据是已经经过“本地计算”的局部最优解，最终的全局最优解只是各个节点返回的局部最优解的一个简单汇总。很显然，这样的设计思路是要充分利用各个节点的并行计算能力，并且避免大量明细数据的网络传输开销。

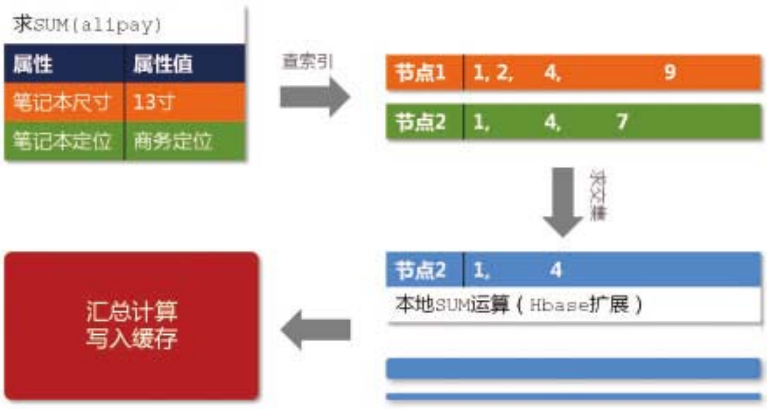


图7 Prom查询过程

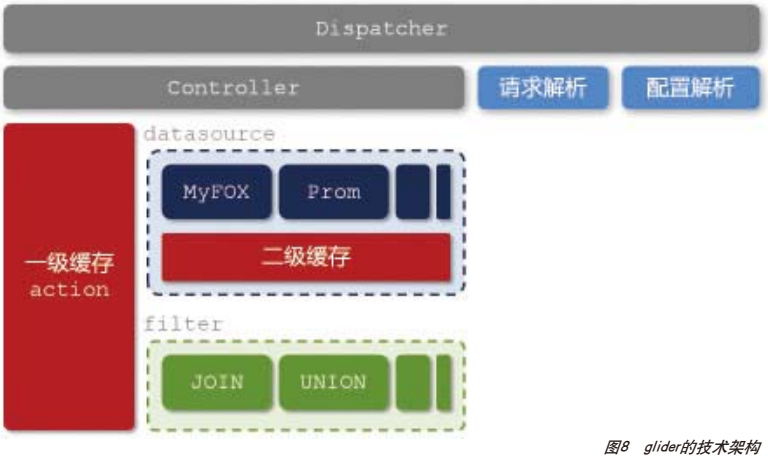


图8 glider的技术架构

用中间层隔离前后端

上文提到过，MyFOX和Prom为数据产品的不同需求提供了数据存储和底层查询的解决方案，但随之而来的问题是，各种异构的存储模块给前端产品的使用带来了很大的挑战。并且，前端产品的一个请求所需要的数据往往不可能只从一个模块获取。

举个例子，我们要在数据魔方中看昨天做热销的商品，首先从MyFOX中拿到一个热销排行榜的数据，但这里的“商品”只是一个ID，并没有ID所对应的商品描述、图片等数据。这个时候我们要从淘宝主站提供的接口中去获取这些数据，然后一一对应到热销排行榜中，最终呈现给用户。

有经验的读者一定可以想到，从本质上来讲，这就是广义上的异构“表”之间的JOIN操作。那么，谁来负责这个事情呢？很容易想到，在存储层与前端产品之间增加一个中间层，它负责各个异构“表”之间的数据JOIN和UNION等计算，并且隔离前端产品和后端存储，提供统一的数据查询服务。这个中间层就是glider（如图8所示）。

缓存是系统化的工程

除了起到隔离前后端以及异构“表”之间的数据整合的作用之外，glider的另外一个不容忽视的作用便是缓存管理。上文提到过，在特定的时间段内，我们认为数据产品中的数据是只读的，这是利用缓存来提高性能的理论基础。

在图8中我们看到，glider中存在两层缓存，分别是基于各个异构“表”（datasource）的二级缓存和整合之后基于独立请求的一级缓存。除此之外，各个异构“表”内部可能还存在自己的缓存机制。细心的读者一定注意到了图3中MyFOX的缓存设计，我们没有选择对汇总计算后的最终结果进行缓存，而是针对每个分片进行缓存，其目的在于提高缓存的命中率，并且降低数据的冗余度。

大量使用缓存的最大问题就是数据一致性问题。如何保证底层数据的变化在尽可能短的时间内体现给最终用户呢？这一定是一个系统化的工程，尤其对于分层较多的系统来说。

图9向我们展示了数据魔方在缓存控制方面的设计思路。用户的请求中一定是带了缓存控制的“命令”的，这包括URL中的query string，和HTTP头中的“If-None-Match”信息。并且，这个缓存控制“命令”一定会经过层层传递，最终传递到底层存储的异构“表”模块。各异构“表”除了返回各自的数据之外，还会返回各自的数据缓存过期时间（ttl），而glider最终输出的过期时间是各个异构“表”过期时间的最小值。这一过期时间也一定是从底层存储层层传递，最终通过HTTP头返回给用户浏览器的。

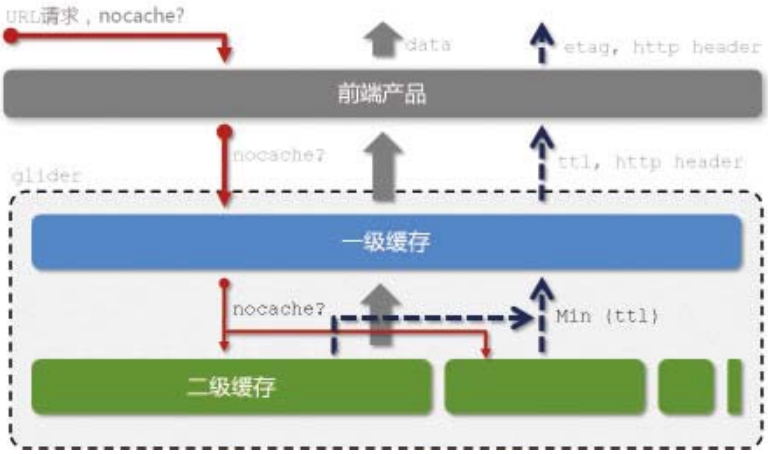


图9 缓存控制体系

缓存系统不得不考虑的另一个问题是缓存穿透与失效时的雪崩效应。缓存穿透是指查询一个一定不存在的数据，由于缓存是不命中时被动写的，并且出于容错考虑，如果从存储层查不到数据则不写入缓存，这将导致这个存在的数据每次请求都要到存储层去查询，失去了缓存的意义。

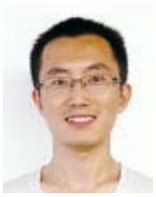
有很多种方法可以有效地解决缓存穿透问题，最常见的则是采用布隆过滤器，将所有可能存在的数据哈希到一个足够大的bitmap中，一个一定不存在的数据会被这个bitmap拦截掉，从而避免了对底层存储系统的查询压力。在数据魔方里，我们采用了一个更为简单粗暴的方法，如果一个查询返回的数据为空（不管是数据不存在，还是系统故障），我们仍然把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。

缓存失效时的雪崩效应对底层系统的冲击非常可怕。遗憾的是，这个问题目前并没有很完美的解决方案。大多数系统设计者考虑用加锁或者队列的方式保证缓存的单线程（进程）写，从而避免失效时大量的并发请求落到底层存储系统上。在数据魔方中，我们设计的缓存过期机制理论上能够将各个客户端的数据失效时间均匀地分布在时间轴上，一定程度上能够避免缓存同时失效带来的雪崩效应。

结束语

正是基于本文所描述的架构特点，数据魔方目前已经能够提供压缩前80TB的数据存储空间，数据中间层glider支持每天4000万的查询请求，平均响应时间在28毫秒（6月1日数据），足以满足未来一段时间内的业务增长需求。

尽管如此，整个系统中仍然存在很多不完善的地方。一个典型的例子莫过于各个分层之间使用短连接模式的HTTP协议进行通信。这样的策略直接导致在流量高峰期单机的TCP连接数非常高。所以说，一个良好的架构固然能够在很大程度上降低开发和维护的成本，但它自身一定是随着数据量和流量的变化而不断变化的。我相信，过不了几年，淘宝数据产品的技术架构一定会是另外的样子。



张轩丞

花名朋春，淘宝网数据平台与产品部技术专家。目前致力于海量数据产品的分布式存储、检索和实时计算研究。脚本语言爱好者，CNode社区组织者之一。曾在百度工作三年，设计并主导开发了百度分布式定时任务调度系统。

数据驱动销售——个性化推荐引擎

■ 文 / 简朝阳

在当前这个信息量飞速增长的时代，一个企业，尤其是电子商务企业的成功已经越来越多地与其海量数据处理能力相关联。高效、迅速地从海量数据中挖掘出潜在价值并转化为决策依据的能力，将成为企业的核心竞争力。

数据的重要性毋庸置疑，但随着数据的产生速度越来越快，数据量越来越大，数据处理技术的挑战自然也越来越大。如何从海量数据中挖掘出价值所在，分析出深层含义，进而转化为可操作的信息，已经成为各互联网企业尤其是电子商务公司不得不研究的课题。本文将介绍国内箱包行业电子商务领军者麦包包如何利用海量数据的分析处理（个性化推荐引擎）来协助用户更好地完成购买体验。

数据层基础架构

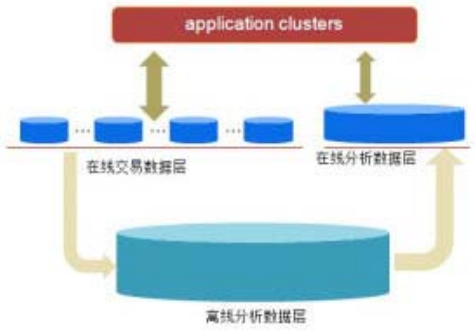


图1　数据层基础架构

如图1所示，麦包包的数据层基础架构与其他很多互联网公司相比，可能会有一点儿差异，那就是有一个用于实时分析处理的在线分析数据层，用来处理一些对实时性要求较高的分析任务。

总的来说，麦包包的数据层分为下面三个部分。

■ 在线交易数据层

用于存放网站对外访问数据，如交易相关、产品相关、用户相关等数据。

■ 离线分析数据层

用于分析各种报表、数据挖掘，如购买行为、销售分析、浏览跟踪等。

■ 在线分析数据层

用于处理一些对实时性要求较高的分析，如在线交易分析、用户浏览推荐等。

在线交易数据层和离线分析数据层对于大家来说都已经比较熟悉了，二者的数据特点和访问特点都很清晰明确，架构方向也相对明确。只有在线分析系统比较特别，既有高并发的用户访问，同时又兼具了分析型复杂查询及海量的基础数据，构建起来相对要复杂一些。所以下面简单介绍一下麦包包如何构建在线分析系统的应用之一——“个性化推荐引擎”。

个性化推荐引擎

我们首先分析一下这个推荐引擎的需求。

■ 关联个性化

根据用户的喜好倾向以及访问历史记录，不同用户浏览同一个产品时，将给出不同的关联推荐结果。

■ 页面个性化

不同用户访问同一个页面，我们将会根据用户的以往购买历史及浏览行为而展示个性化的内容。

■ 搜索个性化

随着用户的多次搜索及结果点击行为，我们会对搜索结果进行过滤重组，尽可能展示更符合用户需求的搜索结果。也就是说，在完全相同的基础数据中，不同用户在同一时间搜索同一个关键词，可能会给出不一样的结果；或者同一个用户重复多次搜索同一个关键词，也可能会有不一样的结果。

我们再来看一看推荐引擎的数据特点。

■ **海量**

超过500万会员，5位数的SKU，7位数的访问量。将这些数据与会员及SKU的各类属性相互关联，数据量之庞大可想而知。

■ **多维度**

从性能优化角度来说，数据量大并不可怕，只要访问方式简单，很容易通过索引等手段进行优化。可偏偏不幸的是，由于将用户和产品进行多维度关联，既需要根据用户去分析，又需要根据产品去关联，再辅以运行时的各类属性；既可能各个维度同时存在，也可能只有任何一个维度；多维度就多维度吧，可还有很多访问是分析型，比较难以优化扩展。

■ **访问高并发**

当然，数据量大也并不一定就可怕，如果并发访问较小，响应时间要求不是太高，那也容易解决，可以用Hadoop之类的分布式系统来分析计算。可恰恰不巧的就是这个系统面对的是网站上的访问客户，对并发及响应时间的要求和OLTP系统一样。

需求已经确定，数据特点也已了解，下一步就是根据数据的特点，设计一个切实可行的架构来实现这些应用需求了。

在如此**海量**数据中进行**高并发**的**复杂分析查询**，还要能够**快速响应**，看上去就像是一个不可能的任务。但仔细分析之后，我们不难发现，推荐引擎结果主要由以下几个因素决定。

■ **用户固定属性**：年龄、性别、职业类型、地域、价格承受范围、色彩喜好、品牌喜好等。

■ **产品固定属性**：品牌、类别、材质、价格、色系等。

■ **用户以往行为**：浏览历史、购买历史等。

■ **用户当前行为**：当前点击、浏览等。

以上四个因素实际上对应了四种数据，在分析每一种数据的特点之后，可以发现前面三个因素所对应的数据都是相对静态的，只有用户当前行为才是一个在不断变化的动态数据。也就是说，在海量数据中，只有少部分数据是动态的，其他大部分都是静态。

当然，用户属性中的各种喜好，也需要我们通过用户以往的历史购买以及浏览行为进行各种分析挖掘才能获得，但这都是由历史积淀数据分析得来，而不是由当前的运行时动态数据决定。价格承受范围以及地域特性也同样如此。

数据的这一特性对我们的架构设计起到了一个非常关键的作用，因为我们可以使用完全不同的方式来将静态数据和动态数据分开处理，再合并分析。静态数据的变化较小，实时性要求较低，我们将进行离线分析；动态数据相对较少，但实时性要求较高，我们在线实时处理。动、静数据在线合并分析。这样一来，我们就可以很轻松地绕过海量数据的高并发在线分析的问题，将这一动作交由离线分析系统定时作业批量完成，既不会有高并发问题，又不存在响应时间的压力。至于在线实时数据的处理，由于数据量的大幅缩减，以及访问方式的简化，比在线交易的OLTP系统复杂度高不了太多，自然也就容易优化了。

架构设计

简单来说，推荐引擎系统本身的基础架构就如图2所展现的一样，一部分数据进行离线计算，另一部分数据在线计算合并，最终通过推荐引擎API将数据处理后返回给前端应用。

看上去简单，但有几个问题并没有展现出

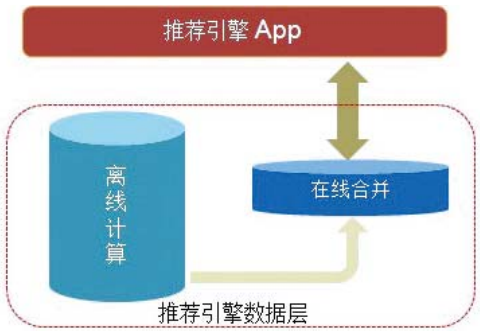


图2 推荐引擎基本架构

来，那就是离线计算和在线计算这两部分具体是如何构建的？数据如何进入离线计算系统？又如何将离线运算结果回送至在线计算系统中？最终数据又如何交由前端应用使用？让我们再来看看图3。

图3 推荐引擎整体架构

离线分析层完全可以通过成熟的产品来构建，如Greenplum、Hadoop等，目前我们已经使用了Greeplum，后续很快还会引入Hadoop，通过HBase + Hive来对处理我们的用户与各SKU的关系数据，帮助进一步完善我们的协同过滤算法，进而优化推荐引擎。

在线合并分析层我们选择MySQL数据库。可能有些人会问，为什么不使用当前如此流行的NoSQL产品呢？主要原因有以下两点。

- MySQL更便于维护与备份等运维需求。
- NoSQL不满足我们的一些分析型查询需求。

NoSQL产品虽然流行，但每种产品都还只适于某些特定的应用场景，很多听上去完美的理论目前暂时还仅仅只是听上去完美，实际用起来仍然存在各种各样的问题。所以我们选择了更适合于我们的MySQL作为在线合并分析层的数据库。

整个架构的数据流，如图3所示。

- 前端应用产生用户的浏览日志、购买日志、搜索日志以及用户及产品属性数据进入。
- 通过文件日志收集程序以及基于MySQL开放复制协议所定制的数据同步工具（注：在我的个人网站上有介绍：http://isky000.com/database/mysql-replication-extend）将数据同步至离线分析系统中。
- 通过离线任务的统计分析，得出会员的各种喜好属性，并将之与产品属性进行关联分析，得出一个用户产品倾向性关联结果，然后再通过应用程序定期从离线分析系统将上述分析结果写入在线合并分析数据库中。
- 推荐引擎根据前端应用（如Search）传入的用户当前运行时操作属性，与在线合并分析数据库中属性进行合并（Merge），再过滤（Filter）。
- 前端应用从推荐引擎处获取Merge与Filter之后的数据，再在前端页面上完成展现。

以上就是整个推荐引擎的数据流架构方案，乍一看也没有太多特别的内容，但在实际实施过程中，会遇到以下几个难点。

- **各种日志传输到离线分析系统，如何做到尽可能实时并不影响在线系统。**

这个难点，我们首先在每一个页面部署监测点，通过请求一个gif图片来获得用户的各种浏览信息，并存入到MySQL数据库，交易相关的数据自然也会有在数据库中进行存储。然后使用通过扩展MySQL复制协议而实现的日志解析合并程序，实时解析 MySQL日志，再将其以我们需要的格式传输至离线分析系统中进行分析运算。

- **如何将用户的运行时操作属性与我们的离线分析结果进行Merge及Filter。**

这个难点，实际上在6月刊的《程序员》杂志对麦包包首席架构师盛国军的采访稿中，已经有了相应的介绍。我们主要使用了基于用户（User）、商品（Item）、话题（Topic）以及曝光（Exposure）这四种协同过滤技术，来实现推荐算法。

总的来说，数据量的增长，以及分析需求的越来越复杂，将会对互联网公司的数据处理能力提出越来越高的要求、越来越大的挑战。但每一个场景都有其特性，充分分析其数据特性，将合适的软件用在合适的场景下，才能更好地解决实际问题。

简朝阳

麦包包技术保障部总监，Oracle ACE，擅长MySQL和Oracle数据库应用系统的性能调优与高可用可扩展架构设计。曾就职于阿里巴巴，参与多个核心数据库应用系统的设计与实施，负责MySQL数据库应用系统的架构设计与相关维护。

视频网站的Big Data解决之道

■ 文 / 姚键

概述

优酷作为一家大型视频网站，拥有海量播放流畅的视频。我们秉承注重用户体验这一产品技术理念，将绝大部分存储用在视频资源上。通过建设专用的视频CDN，建立了可自由扩展、性能优异的架构，在提供更好用户体验的同时优化了存储资源。在除视频资源外的其他方面，我们也累积了海量数据：仅运营数据，每天收集到的网站各类访问日志总量已经达到TB级，经分析及压缩处理后留存下来的历史运营数据已达数百TB，很快将会达到PB级，5年后数据量将会达到几十PB级。

如何更好地处理和分析这些海量数据，以挖掘出其中的价值？

挖掘数据中的价值

对企业来说，尤其是对于为用户提供服务的行业，仅提供基础服务已经越来越难应付日趋细化的商业模式。如何为用户提供差异化的优质服务成为这类企业必须解决的问题。而数据好比灯塔，能为企业指引前进的方向。互联网、电信、金融等行业都在加大数据的探索及应用力度，这为企业创造了可观的经济效益。

对优酷而言，通过用户的每次播放流程，我们都对页面浏览、评论收藏、视频播放以及播放时的各种操作进行了记录。经处理后的分析结果会反馈给不同的业务模块，对包括产品、内容运营、用户的个性化推荐及广告投放等方面的提升，都起到了关键作用。

网站页面、客户端的UI/UE的设计及效果，都需要数据进行支持。通过A/B测试系统，我们收集到用户对不同UI下的操作反馈，进而评估UI的改变对用户的影响。

内容方面，通过对用户网络情况的统计：每次播放是否发生了缓冲，平均下载速度是多少等，进行实时的统计和计算，获取每个地区每个运营商下用户的加载表现，以此来决定CDN节点的分布和分配策略，为不同地区、不同运营商的用户提供清晰流畅的视频服务。

在推荐方面，通过对大量视频播放行为的分析，归纳不同时长、不同类型、不同内容的视频之间的相互关联，挖掘不同人群用户的同质化观看习惯，对每次用户的观看进行有针对性的后续推荐，并借助后续数据的分析，迭代地改善现有服务，为用户提供量身定制的推送服务。

数据对于优酷的广告精准投放也起到了重要作用。优酷的广告系统支持对不同地域、频道、标签及人群等条件的定向投放。在投放策略上，我们本着尽可能不影响用户体验的原则，对于广告长度及投放频次都进行了限制。虽然这给投放造成了较大难度，但通过对各种细粒度的定向条件组合历史数据进行的分析，我们在广告投放方面已经相当准确。在投放阶段，我们实时分析用户属性、访问情况及当前广告投放量，对每次投放动态调整。

技术架构

下面以优酷的运营数据为例介绍我们的海量数据解决之道。我们的运营数据包括播放、用户交互、搜索、广告等，目前总计达数百TB，它们存储在数百台服务器上。我们主要使用的是内部专门开发的轻量级的分布式存储及数据分析框架，应用于一百台左右的服务器集群，目前仍在使用。另外我们搭建了一个1000个Slot的Hadoop集群，并还在继续扩展。考虑到维护成本及扩展性，未来我们会将全部业务迁移到Hadoop平台上，以降低维护成本。另外基于Hadoop及其上层的HBase、Hive等数据存储产品，我们会开发出一套数据处理框架，应用于整个数据处理系统。

如图1所示，根据业务类型的不同，我们收集到日志之后，按照不同时间策略先对数据进行清洗。不失真的原始日志，按规定的格式直接以文件的形式存储在Hadoop上，数据清洗转换后的中间结果，会存储在Hive数据仓库上；而一些粗粒度的汇总数据，则写到MySQL、HBase等数据库中。

每天优酷的日常数据处理任务多达数百个，对时效性要求不同，任务的执行策略也不同。需要准实时查询的，我们可以提供延迟10分钟的数据；其他任务也根据优先级及紧急程度安排调度，而执行中的资源的分配由系统动态调整。

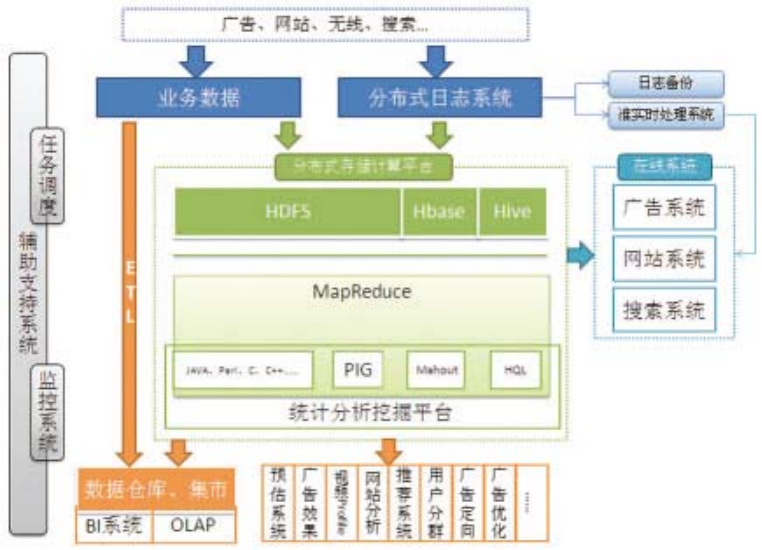


图1　运营数据处理技术架构

NoSQL探索之路

层出不穷的NoSQL技术，无疑是现在极其热门的领域，依托高可用性、高水平扩展性、高效存取及支持MapReduce等特性使其在应对Web2.0网站时比关系型数据库更加得心应手。

目前优酷大量数据依然存储在MySQL等平台上，这是考虑到关系型数据库大都经历了长时间的实践检验，比较成熟，遵循相同标准，能获得较好的支持。而且，主流关系型数据库也都积极尝试从海量数据等方面改进产品。此外，NoSQL分Key-Value、document、column、图等多种类型，特性各不相同，这种总称屏蔽了彼此之间的差异；而且各NoSQL产品还在不断变化，甚至API也有所更改，这在技术掌握上及维护上成本较高，毕竟找一个懂MySQL及SQL的开发工程师比找一个MongoDB专家要容易得多。

尽管如此，仍阻挡不住我们对NoSQL产品进行尝试的热情：一方面对不同业务应用适合的产品，另一方面对同类产品也在做评估对比。目前优酷的在线评论业务已部分迁移到MongoDB，运营数据分析及挖掘处理我们在使用Hadoop/HBase；在Key-Value产品方面，我们也在寻找更优的Memcache替代品，如Redis，相对于Memcache，除了对Value的存储支持三种不同的数据结构外，同一个Key的Value进行部分更新也会更适合一些对Value频繁修改的在线业务；同时我们在搜索产品中应用了Tokyo Tyrant；对于Cassandra等产品我们也进行过研究。可以说，我们会一直持续关注NoSQL技术，未来可能会根据需要选择合适的产品应用到实际业务中。

目前NoSQL技术依然处于发展阶段，只有少数蕴涵优秀的技术，并最终幸免淘汰，与其他数据解决方案一起生存下来，未来也许会有新的NoSQL产品出现。开发人员，尤其是各大公司的技术团队在使用NoSQL项目的同时，根据使用经验，会对一些针对特定使用场景的特性及服务加以抽象并实现，形成类似于基于

Hadoop平台的Pig这类衍生项目。在这方面未来的发展潜力是巨大的，比如在BI领域，目前的NoSQL产品与现有BI产品几乎没有交集。将来一些NoSQL产品应该能够通过现有主流BI产品直接访问，或者提供自己的BI模块。类似的功能也许会形成新的标准，颠覆现有开发模式。

机遇大于挑战

海量数据时代对企业的影响，并非直接来自数据，而在于社会的飞速进步及不断涌现的商业模式。2007年，微软CEO鲍尔默说，Google公司现在几乎是每年翻一番。优酷从起步，到刷新5年多来美国IPO最大涨幅纪录，只用了短短4年时间。而随着变革速度不断加快，各企业掉队落伍的可能性已明显增大。从海量数据中挖掘潜在价值，分析行业趋势，在短时间内更新业务模式，优化产品和服务，提升核心竞争力，将是企业需要持续解决的课题。

随着数据量越来越大，并行计算得到了越来越普遍的应用，新技术的产生使得技术选择多元化，学习成本有所提高；同时，在学习之后的应用阶段，开发、测试及维护成本都大大降低，数据分析、挖掘及BI领域依旧会得到成型软件的支持。因此，海量数据时代的技术人员，机遇反而大于挑战：一方面需要从关注开发细节，转而关注各种新技术的特性；另一方面应该具备更深的业务抽象能力。只有具备这样的综合能力，才能让技术发挥更大的价值。

展望

目前，企业所掌握的数据，还远没有达到最细的粒度，随着技术的发展和设备的廉价化，未来企业能够收集到的数据还将呈非线性增长。能够为运营提供支持的新维度、新指标也会被继续发掘出来。分析的难度随着维度和指标的增多不断加大，传统的数据挖掘及BI产品将越来越难满足需求。

在应对海量数据的计算模型方面，目前MapReduce成为主流，大多数平台和产品都在应用MapReduce。而未来，也许会有更好的或者针对特定领域更好的计算模型出现。目前Google在图处理方面，已在使用Pregel模型；Yahoo!也在开发新一代的MapReduce模型，以期从可靠性、可用性、可扩展性、向后兼容、延迟及集群利用等方面，对现有模型进行改进。这类改进会进一步提高海量数据的计算效率。

并行计算及分布式系统未来的产品线应会不断丰富。风靡一时的Hadoop并非完美，高吞吐量造成的高延迟、处理时的资源浪费等，也都会是使用者需要考虑的问题。会不会有替代品出现，让我们拭目以待。

对于具体企业，不同企业可能选用不同的解决方案。一些中小企业，通过购买SaaS服务，将数据提供给第三方服务商处理。服务商通过对多家同行企业的数据分析，寻找共性，再反馈给企业为其创造更大价值，实现了平台提供者和使用者的双赢。另外一些企业会通过租用PaaS及IaaS服务的方式，将相对不敏感的数据放到公有云上处理，以节约设备采购及维护成本。对于企业的敏感数据，会通过建立私有数据中心，搭建私有云来处理，而同时也会考虑用公有云平台进行互补。最终结果，数据会向一些大的节点集中，而大的节点也更有能力从掌握的海量数据中挖掘有用信息。这些信息通用性强的部分会被共享出来，服务于社会。

对于优酷来说，仍处于飞速发展阶段，已经在考虑未来自建数据中心，提高数据处理能力，从网站的运营中发掘出更多信息，为用户提供更好的视频服务。

姚键

优酷CTO。2006年加入优酷，带领技术团队自建CDN网络，同时在产品设计、WEB开发、底层研发、广告系统、无线业务、运营维护等方面全面发力。曾在ChinaRen、搜狐、新东方在线、51Credit等互联网站任技术管理职位。

大数据下的数据分析平台架构

■ 文 / 谢超

随着互联网、移动互联网和物联网的发展，谁也无法否认，我们已经切实地迎来了一个海量数据的时代，数据调查公司IDC预计2011年的数据总量将达到1.8万亿GB，对这些海量数据的分析已经成为一个非常重要且紧迫的需求。

作为一家互联网数据分析公司，我们在海量数据的分析领域那真是被“逼上梁山”。多年来在严苛的业务需求和数据压力下，我们几乎尝试了所有可能的大数据分析方法，最终落地于Hadoop平台之上。

Hadoop在可伸缩性、健壮性、计算性能和成本上具有无可替代的优势，事实上已成为当前互联网企业主流的大数据分析平台。本文主要介绍一种基于Hadoop平台的多维分析和数据挖掘平台架构。

大数据分析的分类

Hadoop平台对业务的针对性较强，为了让你明确它是否符合你的业务，现粗略地从几个角度将大数据分析的业务需求分类，针对不同的具体需求，应采用不同的数据分析架构。

■ **按照数据分析的实时性，分为实时数据分析和离线数据分析两种**。实时数据分析一般用于金融、移动和互联网B2C等产品，往往要求在数秒内返回上亿行数据的分析，从而达到不影响用户体验的目的。要满足这样的需求，可以采用精心设计的传统关系型数据库组成并行处理集群，或者采用一些内存计算平台，或者采用HDD的架构，这些无疑都需要比较高的软硬件成本。目前比较新的海量数据实时分析工具有EMC的Greenplum、SAP的HANA等。

对于大多数反馈时间要求不是那么严苛的应用，比如离线统计分析、机器学习、搜索引擎的反向索引计算、推荐引擎的计算等，应采用离线分析的方式，通过数据采集工具将日志数据导入专用的分析平台。但面对海量数据，传统的ETL工具往往彻底失效，主要原因是数据格式转换的开销太大，在性能上无法满足海量数据的采集需求。互联网企业的海量数据采集工具，有Facebook开源的Scribe、LinkedIn开源的Kafka、淘宝开源的Timetunnel、Hadoop的Chukwa等，均可以满足每秒数百MB的日志数据采集和传输需求，并将这些数据上载到Hadoop中央系统上。

■ **按照大数据的数据量，分为内存级别、BI级别、海量级别三种**。

这里的内存级别指的是数据量不超过集群的内存最大值。不要小看今天内存的容量，Facebook缓存在内存的Memcached中的数据高达320TB，而目前的PC服务器，内存也可以超过百GB。因此可以采用一些内存数据库，将热点数据常驻内存之中，从而取得非常快速的分析能力，非常适合实时分析业务。图1是一种实际可行的MongoDB分析架构。

图1　用于实时分析的MongoDB架构

MongoDB大集群目前存在一些稳定性问题，会发生周期性的写堵塞和主从同步失效，但仍不失为一种潜力十足的可以用于高速数据分析的NoSQL。

此外，目前大多数服务厂商都已经推出了带4GB以上SSD的解决方案，利用内存+SSD，也可以轻易达到内存分析的性能。随着SSD的发展，内存数据分析必然能得到更加广泛的应用。

BI级别指的是那些对于内存来说太大的数据量，但一般可以将其放入传统的BI产品和专门设计的BI数据库之中进行分析。目前主流的BI产品都有支持TB级以上的数据分析方案。种类繁多，就不具体列举了。

海量级别指的是对于数据库和BI产品已经完全失效或者成本过高的数据量。海量数据级别的优秀企业级产品也有很多，但基于软硬件的成本原因，目前大多数互联网企业采用Hadoop的HDFS分布式文件系统来存储数据，并使用MapReduce进行分析。本文稍后将主要介绍Hadoop上基于MapReduce的一个多维数据分析平台。

■ 数据分析的算法复杂度

根据不同的业务需求，数据分析的算法也差异巨大，而数据分析的算法复杂度和架构是紧密关联的。举个例子，Redis是一个性能非常高的内存Key-Value NoSQL，它支持List和Set、SortedSet等简单集合，如果你的数据分析需求简单地通过排序，链表就可以解决，同时总的数据量不大于内存（准确地说是内存加上虚拟内存再除以2），那么无疑使用Redis会达到非常惊人的分析性能。

还有很多易并行问题（Embarrassingly Parallel），计算可以分解成完全独立的部分，或者很简单地就能改造出分布式算法，比如大规模脸部识别、图形渲染等，这样的问题自然是使用并行处理集群比较适合。

而大多数统计分析，机器学习问题可以用MapReduce算法改写。MapReduce目前最擅长的计算领域有流量统计、推荐引擎、趋势分析、用户行为分析、数据挖掘分类器、分布式索引等。

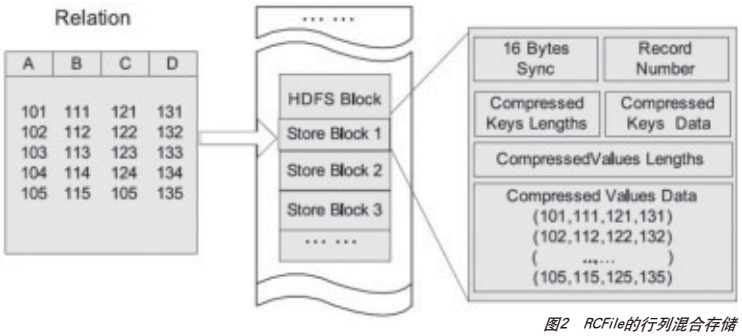


图2 RCFile的行列混合存储

面对大数据OLAP分析的一些问题

OLAP分析需要进行大量的数据分组和表间关联，而这些显然不是NoSQL和传统数据库的强项，往往必须使用特定的针对BI优化的数据库。比如绝大多数针对BI优化的数据库采用了列存储或混合存储、压缩、延迟加载、对存储数据块的预统计、分片索引等技术。

Hadoop平台上的OLAP分析，同样存在这个问题，Facebook针对Hive开发的RCFile数据格式，就是采用了上述的一些优化技术，从而达到了较好的数据分析性能。如图2所示。

然而，对于Hadoop平台来说，单单通过使用Hive模仿出SQL，对于数据分析来说远远不够，**首先**Hive虽然将HiveQL翻译为MapReduce的时候进行了优化，但依然效率低下。多维分析时依然要做事实表和维度表的关联，维度一多性能必然大幅下降。**其次**，RCFile的行列混合存储模式，事实上限制死了数据格式，也就是说数据格式是针对特定分析预先设计好的，一旦分析的业务模型有所改动，海量数据转换格式的代价是极其巨大的。**最后**，HiveQL对OLAP业务分析人员依然是非常不友善的，维度和度量才是直接针对业务人员的分析语言。

而且目前OLAP存在的最大问题是：业务灵活多变，必然导致业务模型随之经常发生变化，而业务维度和度量一旦发生变化，技术人员需要把整个Cube（多维立方体）重新定义并重新生成，业务人员只能在此Cube上进行多维分析，这样就限制了业务人员快速改变问题分析的角度，从而使所谓的BI系统成为死板的日常报表系统。

使用Hadoop进行多维分析，首先能解决上

述维度难以改变的问题，利用Hadoop中数据非结构化的特征，采集来的数据本身就是包含大量冗余信息的。同时也可以将大量冗余的维度信息整合到事实表中，这样可以在冗余维度下灵活地改变问题分析的角度。其次利用Hadoop MapReduce强大的并行化处理能力，无论OLAP分析中的维度增加多少，开销并不显著增长。换言之，Hadoop可以支持一个巨大无比的Cube，包含了无数你想到或者想不到的维度，而且每次多维分析，都可以支持成千上百个维度，并不会显著影响分析的性能。

因此，我们的大数据分析架构在这个巨大Cube的支持下，直接把维度和度量的生成交给业务人员，由业务人员自己定义好维度和度量之后，将业务的维度和度量直接翻译成MapReduce运行，并最终生成报表。可以简单理解为用户快速自定义的“MDX”（多维表达式，或者多维立方体查询）语言→MapReduce的转换工具。同时OLAP分析和报表结果的展示，依然兼容传统的BI和报表产品。如图3所示。

由图3可以看出，在年收入上，用户可以自己定义子维度。另外，用户也可以在列上自定义维度，比如将性别和学历合并为一个维度。由于Hadoop数据的非结构化特征，维度可以根据业务需求任意地划分和重组。

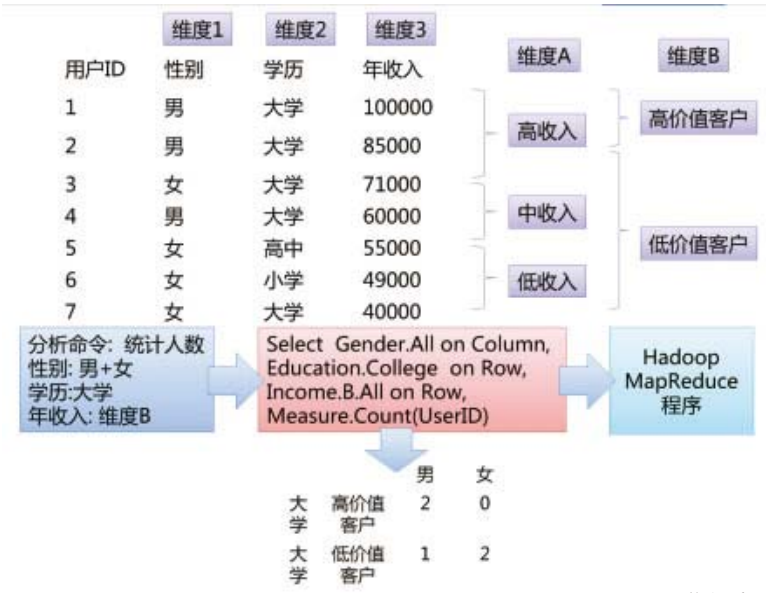


图3 MDX→MapReduce简略示意图

图4 Hadoop多维分析平台架构图

一种Hadoop多维分析平台的架构

整个架构由四大部分组成：数据采集模块、数据冗余模块、维度定义模块、并行分析模块。如图4所示。

数据采集模块采用了Cloudera的Flume，将海量的小日志文件进行高速传输和合并，并能够确保数据的传输安全性。单个collector宕机之后，数据也不会丢失，并能将agent数据自动转移到其他的colllecter处理，不会影响整个采集系统的运行。如图5所示。

数据冗余模块不是必须的，但如果日志数据中没有足够的维度信息，或者需要比较频繁地增加维度，则需要定义数据冗余模块。通过冗余维度定义器定义需要冗余的维度信息和来源（数据库、文件、内存等），并指定扩展方

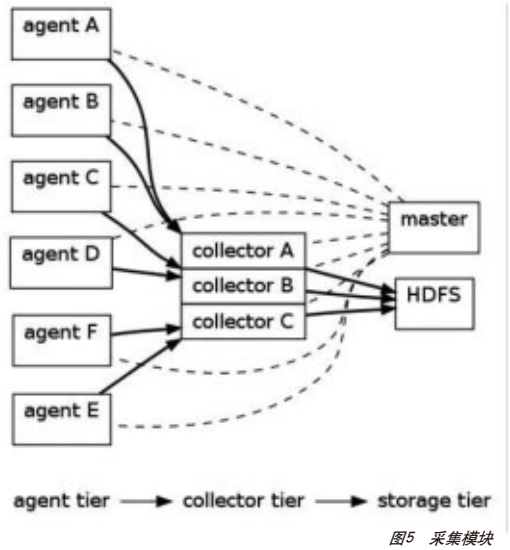


图5 采集模块

图6 核心模块的逻辑

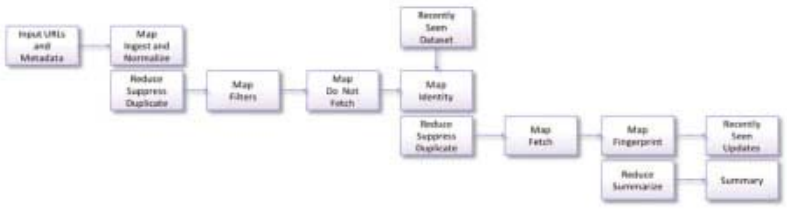


图7 MapReduce WorkFlow例子

式，将信息写入数据日志中。在海量数据下，数据冗余模块往往成为整个系统的瓶颈，建议使用一些比较快的内存NoSQL来冗余原始数据，并采用尽可能多的节点进行并行冗余；或者也完全可以在Hadoop中执行批量Map，进行数据格式的转化。

维度定义模块是面向业务用户的前端模块，用户通过可视化的定义器从数据日志中定义维度和度量，并能自动生成一种多维分析语言，同时可以使用可视化的分析器通过GUI执行刚刚定义好的多维分析命令。

并行分析模块接受用户提交的多维分析命令，并将通过核心模块将该命令解析为MapReduce，提交给Hadoop集群之后，生成报表供报表中心展示。

核心模块是将多维分析语言转化为MapReduce的解析器，读取用户定义的维度和度量，将用户的多维分析命令翻译成MapReduce程序。核心模块的具体逻辑如图6所示。

图6中根据JobConf参数进行Map和Reduce类的拼装并不复杂，难点是很多实际问题很难通过一个MapReduce Job解决，必须通过多个MapReduce Job组成工作流（WorkFlow），这里是最需要根据业务进行定制的部分。图7是一个简单的MapReduce工作流的例子。

MapReduce的输出一般是统计分析的结果，数据量相较于输入的海量数据会小很多，这样就可以导入传统的数据报表产品中进行展现。

结束语

当然，这样的多维分析架构也不是没有缺点。由于MapReduce本身就是以蛮力去扫描大部分数据进行计算，因此无法像传统BI产品一样对条件查询做优化，也没有缓存的概念。往往很多很小的查询需要“兴师动众”。尽管如此，开源的Hadoop还是解决了很多人在大数据下的分析问题，真可谓是“功德无量”。

Hadoop集群软硬件的花费极低，每GB存储和计算的成本是其他企业级产品的百分之一甚至千分之一，性能却非常出色。我们可以轻松地进行千亿乃至万亿数据级别的多维统计分析和机器学习。

6月29日的Hadoop Summit 2011上，Yahoo!剥离出一家专门负责Hadoop开发和运维的公司Hortonworks。Cloudera带来了大量的辅助工具，MapR带来了号称三倍于Hadoop MapReduce速度的并行计算平台。Hadoop必将很快迎来下一代产品，届时其必然拥有更强大的分析能力和更便捷的使用方式，从而真正轻松面对未来海量数据的挑战。

谢超

Admaster数据挖掘总监，云计算实践者，10年数据仓库和数据挖掘咨询经验，现专注于分布式平台上的海量数据挖掘和机器学习。

海量空间数据库建设策略

■ 文 / 刘锋

概述

空间数据库是存储在通用文件系统文件夹、Microsoft Access数据库或多用户关系DBMS（如Oracle、SQL Server、PostgreSQL、Informix或DB2）中的各种类型地理数据集的集合，是实现地理信息系统的基础，决定了客观世界中的空间实体将以何种方式在计算机系统中被抽象、存储和管理。由于传统的关系数据库在空间数据的表示、存储、管理、检索上存在许多缺陷，ArcGIS引入一种全新的面向对象的空间数据模型，即建立在DBMS之上的统一的、智能的空间数据模型——Geodatabase。Geodatabase搭建了一个框架，使用户可以轻易地创建智能化空间要素，模拟真实世界中地理对象之间的作用和行为。

海量空间数据库建设的目标

当空间数据库中存储的数据达到一个数量级以后，很多在小数据量情况下可以忽略的问题会随着量变达到一个超出容忍范围的质变，“海量”的称谓也可以被认为是这个量变的临界值。

我们当然不希望数据量的增加带来灾难性的后果，然而，我们也不认为数据库的能力是无穷无尽的。因此，在理想和现实之间，就有了这样一个平衡点：希望在可以预见的数量级内，我们的空间数据库可以保持一个较好的工作状态，数据的存储、维护、索引、查询等操作都可以满足应用的需求。

海量空间数据库建设的目标，是保证在数据量增长到一定程度后，空间数据库还能很好地为应用服务。

Geodatabase的选择

在ArcGIS产品线中，Geodatabase有三种存在形式：Personal Geodatabase、File Geodatabase和ArcSDE。

Personal Geodatabase只支持Windows平台、容量有2GB的限制，数据达到200MB以上性能会下降，因此不适合海量数据的存储。

File Geodatabase每个数据集支持最大1TB的容量，同时允许多个用户的读取。虽然限制了在某个数据集上同时只能有一个用户进行编辑，但对于只读的场景而言，File Geodatabase无疑也是一种选择，毕竟，以文件形式存储的空间数据库，在管理和迁移上要简单得多。

ArcSDE通过与Oracle、PostgreSQL、DB2、Informix、SQL Server等企业级数据库的结合，有几乎无限制的能力。但并非任何情况都采用ArcSDE是最佳选择，技术方案的选择总是伴随着各种因素的反复权衡。

海量空间数据库构建的基本策略

硬件策略

硬件是所有项目的基础，硬件的好坏直接决定着后续的所有策略。硬件策略指的是数据库服务器和应用服务器的硬件配置，这些硬件主要包括如表1所示的几个方面。

软件策略

如果把机器硬件作为人体的骨骼，那软件就是人体的血液了，相对于硬件的策略，软件的调整策略更多、更灵活。在数据库服务器上一般需要考虑以下几种软件：操作系统

表1 硬件测试的指标

Servers	Server type	Configuration	RAM	Required concurrent users	Max capacity (users)	Map Images per second
GDB Minimum	Unix IBM P690	IBM P690 4 core (2 chip) 1900MHz	8GB	40-60	248	28
GDB Maximum	Unix IBM P690	IBM P690 8 core (2 chip) 1900MHz	16GB	40-60	495	54
App.Server Minimum	Windows HP Proliant 580	HP DL580 (Intel Xeon) 2 core(2 chip) 3200MHz	4GB	40-60	16	3
App.Server Maximum	Windows HP Proliant 580	HP DL580 (Intel Xeon) 4 core(2 chip) 3200MHz	8GB	40-60	40	5
App.Server Minimum	Windows IBM x3755(#6)	IBM System X 3755 (AMD Opteron 8222 SE) 2 core(2 chip) 3000MHz	4GB	40-60	54	4
App.Server Maximum	Windows IBM x3755(#6)	IBM System X 3755 (AMD Opteron 8222 SE) 4 core(2 chip) 3000MHz	8GB	40-60	110	8
App.Server Minimum	Windows IBM x3850	IBM System x (Intel Xeon E7210) 2 core (2 chip) 2400MHz	4GB	40-60	32	4
App.Server Maximum	Windows IBM x3850	IBM System x (Intel Xeon E7210) 4 core (2 chip) 2400MHz	8GB	40-60	65	8

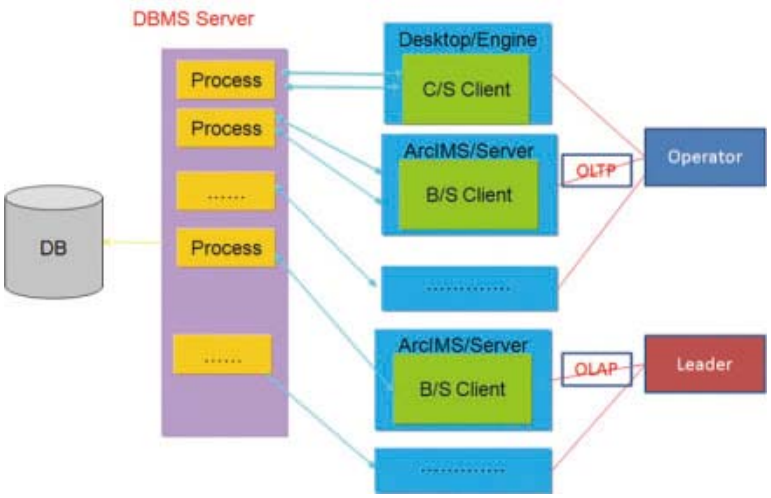


图1 包括两个业务的系统结构图

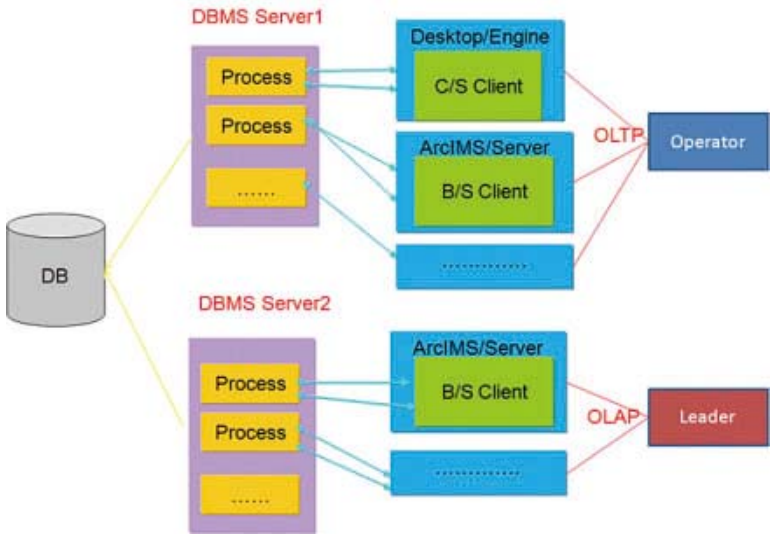


图2 集群的系统结构图

（OS）、数据库（DBMS）、空间数据库中间件（ArcSDE）。应用服务器需要考虑以下几种软件：操作系统（OS）、ArcGIS Server（或者ArcIMS）、WebServer。

操作系统策略

在操作系统层面上主要考虑以下几方面问题。

■ 数据库服务器上计算内存和文件内存的分配大小

在非Windows操作系统上可以将内存分为计算内存和文件内存，并且可以控制这两部分内存的使用情况，根据项目的不同要求需要仔细考虑这两种内存的分配情况。

■ 分页策略

在Unix和Linux操作系统上需要分配好交换空间的大小以及内存的使用情况，尽量避免大量页交换情况的发生。

■ 应用程序之间所占内存的比例

需要规划好数据库服务器上各个应用程序之间的内存使用情况，如果数据库服务器的硬件足够强大，可以考虑将ArcSDE和数据库安装在一起，反之可以分开部署。如果部署在一起的话，需要考虑ArcSDE和数据库所占有的内存比例，避免ArcSDE和数据库抢占系统资源。

数据库集群策略

大部分使用空间数据库的业务都是OLAP/OLTP混合业务，一个包括两个业务的系统结构如图1所示。

如果碰到上述情况可以使用集群策略来解决效率问题，集群的系统结构如图2所示。

图2为两台数据库集群的情况，可以将操作员的所有请求交由第一台数据库服务器来处理，领导所发送的请求交由第二台数据库服务器来处理，这两台机器获得的数据从一个库中所获取，因此可以解决OLAP和OLTP的冲突问题。

ArcSDE部署策略

ArcSDE一共有三种部署方式，如图3所示。

ArcSDE支持两种连接方式——两层直连和三层的应用服务连接，其中三层应用服务连接在部署上又分为两种情况：第一种情况是ArcSDE的服务和数据库部署在一台机器；第二种情况是ArcSDE单独部署在一台机器上。现在ArcGIS推荐的连接方式为直连，因为直连需要每台前端的机器上安装数据的客户端。直连比较适合于B/S架构，因为B/S架构只需要在安装ArcGIS Server的机器上安装客户端就可以了。对于C/S架构，需要视实际情况而定。

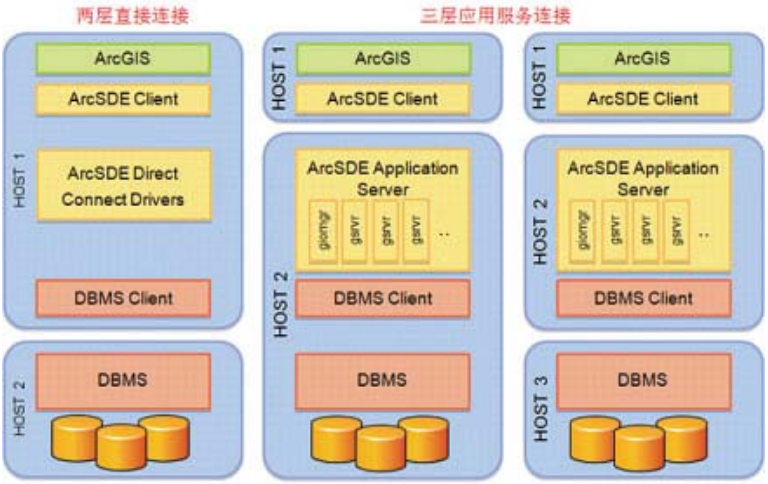


图3　ArcSDE的三种部署方式

空间数据模型分类

我们可以用三种基本方式来模拟地理数据：以矢量格式的离散数据集、以具有光谱或属性数据的像元格网和一系列三角形点拟合一个表面。

矢量数据模型

以矢量方式组织数据、用于对实际地理空间的现象和特征进行模拟和演示的数据模型。矢量数据反映的是点、线、多边形要素，常应用于具有确定形状或边界的不连续对象。要素具有精确的形状和位置、属性和元数据以及有意义的行为。在Geodatabase中，有多种方式可以组织矢量数据，包括Feature Class、Network等。

栅格数据模型

栅格数据表现为连续的数据，栅格中的每一个像元（或像素）是一个测量单元。栅格数据集最常见的来源是卫星影像或航空照片，栅格数据集也可以是一个要素（如建筑物）的照片。

在Geodatabase中，有4种方式可以组织栅格数据：Raster Dataset、Raster Catalog、Mosaic Dataset和Feature的栅格字段。其中，Feature的栅格字段相当于将一个栅格对象作为属性存储到表中，因此并不能作为真正的栅格数据的存储方式。

三角网数据模型

不规则三角网（TIN）以数字方式来表示表面形态，是基于矢量的数字地理数据的一种形式，通过将一系列折点（点）组成三角形来构建。各折点通过由一系列边进行连接，最终形成一个三角网。TIN是描述地块表面的有效方式，在模拟分水岭、可见度、视线、坡度、坡向、山脊和河流以及具有体积的形体时非常有用，它也能模拟点、线和多边形。一个三角形由很多聚点（mass points）组成，每个点具有x、y、z坐标。

下面，将以最常用的矢量和栅格数据库为主，解释海量空间数据库的构建策略。

海量矢量数据库构建策略

数据库配置参数

配置数据参数根据业务来定，如针对于Oracle数据一般需要调整的参数DB_BLOCK_SIZE，针对矢量数据默认的8KB大小已经足以，若是影像数据可以考虑换成16KB或者32KB。

数据组织策略

在数据组织模式上，主要分为按照小单位组织、按照中层单位组织、一张图这三种。如果按照小单位组织和按照中层单位组织的话，在某些功能上前端应用需要开发过多的程序才能完成。推荐一张图的组织方式，http://download.csdn.net/source/3116677中的视频是全国一张图的数据，该图层总共有1亿多条记录，从中可以看出速度。

矢量数据存储类型策略

ArcSDE针对不同的数据支持不同的矢量存储类型，图4是每个数据库所支持的存储类型。

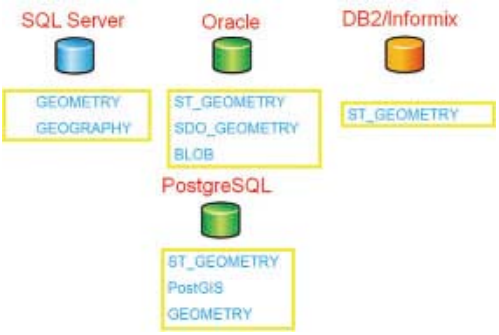


图4 每个数据库所支持的存储类型

不同存储类型比较大的一个区别是所占有的存储空间不同，其中以ST_GEOMETRY存储类型所占有的空间最小，次之是SDELOB，最后才是Oracle的SDO_GEOMETRY。实际上三种存储方式在Oracle的后台都是BLOB进行存储的，只不过ST_GEOMETRY和SDELOB方式对存储的二进制数据ArcGIS进行了压缩和加密处理。

BLOB存储策略

该策略只针对于Oracle数据库，既然在Oracle数据库中后台的存储都是BLOB方式存储，我们推荐BLOB的存储策略为：将BLOB列和常规数据类型的列放在一个块中，前提是BLOB的大小小于3192bytes，并且BLOB的ENABLESTORAGE IN ROW属性在ON的情况下，如图5所示。

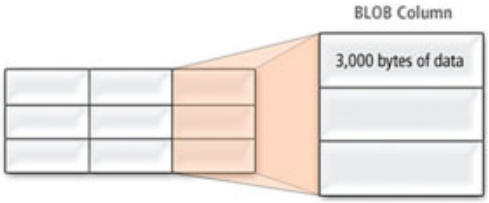


图5 BLOB的存储策略

3192个Bytes能存储约600个点，而大部分数据不会超过600个点串。所以为了实现这一点，建议把ENABLESTORAGE IN ROW选项打开。BLOB存储还有以下几个比较重要的控制参数。

- CACHE参数，用来控制读取到的BLOB是否先放到Oracle的SGA区域中，默认该选项是打开，我们也推荐打开这个选项。
- CHUNK大小，是BLOB分配大小的单位。

表空间分离策略

ArcSDE在存储FeatureClass时，是将相关信息分别存储在B表和S表中，其中B表存储实际数据，而S表存储索引信息，因此最好是将两个表分别存储到不同的介质上这样比较合理一些。

索引策略

索引分为属性索引和空间索引：对于属性索引很多人已经非常熟悉了，如何评估所建立的空间索引是否合理呢？针对Oracle数据库，ArcSDE提供了sdelayer –o si_stats来对已经建立好的索引信息进行统计。

aspx

对DB2数据库，db2 spatial extender提供了工具gseidx，它不但可以统计已建立索引的信息，而且还可以在没建立网格索引的时候统计各种网格大小的统计值。针对完成后的统计信息，可以按照下面的标准来优化你的网格索引。

- 网格的记录数除以地物的记录数要小于2。
- 一个网格中所包括的地物的个数适当低一些，在100~300个是最好的，一个网格中包括的地物的最大个数最好不超过4000。

Cache策略

这一策略针对的数据库是Oracle，包括两方面内容的Cache。

- Cache1策略：可以将SDE的对象Cache到内存中，首先以sys as sysdba登录到数据库中，从DBA_TABLES数据字典中查找出sde schema中所有的上述对象。
- Cache2策略：索引表缓存，按照存储类型不同缓存的表也不相同。

海量栅格数据库构建的主要方式

常用的三种海量影像数据存储的方式

栅格数据集最常见的来源是卫星影像或航空照片，以下简称影像。海量影像数据存储的三种方式是：集中式文件系统存储管理、基于传统关系数据库存储管理、文件与数据库结合的存储管理。

■ **集中式文件系统存储管理模式**

特点是将海量影像数据全部存储于中心服务器上，通过文件的方式对影像数据进行组织管理，文件系统能够很好地使用各种复杂的影像数据结构，并能高效地支持数据的操作和维护，但是系统的可扩展性不强，中心服务器容易成为整个系统的瓶颈。

优点包括：数据传输快；存储容量大。

缺点包括：通用存储；没有针对影像数据管理优化、不易实现对影像的查找。

■ **基于传统关系数据库的存储管理模式**

分布式数据库存储管理包括两种主要形式：第一，对关系型数据库进行扩展，通过增加空间数据类型、空间关系和操作，实现对影像数据的存储和管理；第二，通过构建数据管理中间件（称为空间数据引擎），提供高效的空间数据访问接口，实现对影像数据的存储管理。后者具有良好的安全性、多用户并发访问特性和数据一致性，但系统的存储受制于所依赖的DBMS的能力，并且传统的关系型DBMS对影像数据等非结构化数据的存储效率并不高，对影像数据的管理与发布的支持较弱，从而影响整个系统的性能。

优点包括：成熟的RDBMS技术（安全、多用户访问、并发控制）；针对影像数据管理设计专门的方案；支持影像的查询检索，支持影像数据访问。

缺点包括：影像存储能力受限于数据库的能力；影像数据需要预先上传到数据库中；数据库昂贵。

■ **文件与数据库结合的存储管理模式**

分布式文件系统存储管理的特点是整个系统部署在由通用型桌面计算设备构建的大规模集群之上，将影像数据存储在分布式文件系统的各个数据节点上，中心接到后通过文件的方式对影像数据进行组织管理。该方式的优点是能够很好地适应各种复杂的影像数据结构，高效地支持数据的操作，充分利用了集群中各个节点的计算能力和存储能力，整个系统具有平滑的扩展能力和高可用性。

优点包括：减少数据复制，充分利用文件存储系统的优势；数据库利用率高，充分利用数据库；支持影像检索查询。

缺点包括：不能直接访问影像数据；不能在一张图上显示影像（波段数不同、像元类型不同）；集成性有待增强。

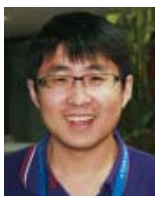
ArcGIS存储和管理影像的数据模型

ArcGIS存储和管理影像的数据模型包括：栅格数据集Raster dataset、栅格目录Raster catalog（9.0）、镶嵌数据集Mosaic dataset。其中Mosaic dataset是伴随ArcGIS 10推出的，专为海量影像数据管理优化的数据模型。它的主要特点有以下这些方面。

- 文件和数据库结合。
- 支持服务共享。
- 快速的编目管理。
- 支持几乎所有格式的影像数据文件。
- 常用传感器影像。
- 在空间数据库中建立影像索引。
- 不会拷贝或改变原有的影像数据。
- 敏捷的元数据管理。
- ISO标准/属性字段。
- 集成的数据访问。
- 作为影像：动态镶嵌、实时处理、自动匀光匀色。
- 作为目录：覆盖区、元数据。

结束语

随着GIS的发展及其应用领域需求的不断扩大，特别是数字地球概念的提出，使得空间数据量也随之剧增。本文围绕如何有效地对海量空间数据进行存储、管理，结合ArcGIS产品和技术特点来满足诸如存储量、访问速度、实时显示等方面的要求，提出了完整的海量空间数据库实施策略。



刘锋

Esri中国（北京）有限公司客户及合作伙伴部高级GIS技术分析师，精通ArcGIS产品系列中的ArcSDE组件和Oracle数据库，主要负责ArcSDE组件的技术支持和产品培训等工作。曾任职于方正电子有限公司。

数据可视化之美——《纽约时报》的一天

■ 文 / Michael Young，Nick Bilton　译 / 祝洪凯，李妹芳

你是否曾经想过《纽约时报》网站的读者会涵盖什么类型的人？我们想过。我们还在想他们倾向于在一天之中的什么时候来访问网站，使用什么工具访问以及他们都来自哪里？从他们是谁到在什么时候、以什么方式以及为什么等，所有这些问题都在我们的思考范围之内。

本文将要介绍的这个可视化项目源于在《纽约时报》研发试验室一次午餐时就这个话题开展的一次简单讨论。正如你将看到的，从非常简单的基于地理的数据集合开始，很快就深入到海量数据和潜在可视化。最终，我们创建了一个可视化用于显示每天《纽约时报》Web站点和手机站点在世界和美国的流量。

收集一些数据

为了对Web站点和手机站点24小时的流量进行可视化，我们需要创建一个可以从《纽约时报》的访问日志中抽取和清洗数据的程序。考虑到我们想要创建一个可以显示在一天内网站的访问次数的可视化并且是一个基于地理信息进行展示的可视化，我们需要的数据包括：

- 在24小时内，用户每次访问Web站点或手机站点的时间戳。
- 每个用户每次访问时所处位置的经度和纬度。

原始的访问日志包含了人们访问Web站点和手机站点的很多有用的信息（比如每个访问者使用什么浏览器）；但其中有很多信息对我们而言是没有用的，因此需要把它们从日志信息中过滤掉。此外，日志中并不包括每个用户每次访问时的经纬度信息，因此这是我们在日志“清洗”过程中需要添加的信息。

《纽约时报》Web站点月独立访问读者约2000万。这意味着，在任何一天Web站点和手机站点上都有几百万次的页面浏览（或点击）；这是我们准备为可视化收集的基础数据。

数据清洗

对于可视化以及其他日志数据的分析，我们只对来自人们的在Web站点和手机站点的点击数感兴趣——而不是来自网络爬虫、机器人或抓取程序。为了过滤这些不必要的数据，我们实现了一段Java代码用于识别出非人工的访问日志并将其从日志中删除。

每天Web站点原始的日志数据访问量大约有500MB～700MB（压缩格式），手机站点的访问量约80MB～100MB（压缩格式）。在数据清洗过程中，我们执行了IP到经纬度的转换，从而得到每个访问用户的精确位置。原始访问日志中已经包含了用户的IP地址，然后我们使用商业数据库把IP转换成地理位置信息。有很多公司提供GeoIP（地理位置IP）数据库，用于实现该转换。

一旦数据被清洗完毕并准确地进行了地理位置编码，只需要对数据再做最后一轮处理。由于原始的访问日志的收集、存储和清理方式，新清洗完的数据是存放在多个文件中的，需要对它们排序之后合并到一个结果文件中去，该文件将包含可视化所需的数据，即一天访问数据。

每天“清洗”后的网站日志数据被存储到360个文件中，每个文件大小约30MB～40MB（压缩格式）。由于每行中增加了一些额外的字段，如GeoIP 信息，“清洗”后的日志文件要大于原始文件。手机站点数据集小得多，因此清洗后的数据存储在一个文件中，大约70MB（压缩格式）。我们每天需要整理当天的每个清洗后的日志文件，并创建按照对Web站点和手机站点的访

问时间戳以及访问者所在的经纬度排序的单个文件（Web站点和手机站点分别生成一个文件）。

Python、Map/Reduce和Hadoop

我们用Python语言创建了一个简单的map/reduce脚本，可以从清洗后的日志文件中过滤掉所有不需要的数据，并输出以逗号作为分隔符的数据，最后还会对数据进行排序。我们使用Amazon的弹性MapReduce Web服务，它允许我们在很多基于Hadoop的EC2的运行实例中运行Python实现的map/reduce。Amazon的EC2运行实例的“配置”不同（低配、中配和高配），不同的配置会提供不同的RAM、CPU核数和内存，因此我们在很多EC2实例中试验运行map/reduce代码，从而找到性价比最好的配置。数据处理需要约10～20分钟（价值几美元），具体所耗时间会依赖于机器的数量（我们从4～10台都尝试了一遍）和EC2实例的配置（我们尝试了低配和中配）。

map/reduce（Hadoop）Job的输出结果是很多有序的文件，这些文件保存在Amazon的S3桶（buckets）中。为了在可视化中把数据放到一个文件中（与前述方式相同，Web站点和手机站点分别存储，各自有一个独立文件），我们从S3下载结果文件到本地，然后按照传统的方法进行排序和归并。现在，数据已经按照期望的方式保存在一个文件中了，可视化的准备工作已经完成。

可视化的第一步

我们在可视化上做的第一个尝试是创建了一个简单的世界地图，将一天之中对《纽约时报》Web站点的每次访问用一个小的黄色圆圈表示，对移动站点的每次访问用一个小的蓝色圆圈表示。除了全球范围的视图，我们还希望创建一个聚焦（或缩放）于美国的视图。

Processing

Processing（面向设计的开源编程语言和集成开发环境）被选作可视化工具，有几个原因。首先，《纽约时报》研发小组中有些人已经有使用Processing完成小的数据可视化的项目经验，他们还拥有使用传感器作为数据收集设备进行探索的经验。此外，我们都是Ben Fry、Casey Reas（Processing创始人）和Aaron Koblin使用该工具所创造的作品的超级粉丝，我们认为Processing将会成为对海量数据进行可视化的理想工具。

对于该可视化，我们需要做的第一件事是将网站的访问用户的经纬度信息映射到Processing中的二维可视化图形中。Aaron Koblin友情提供了一些他在前一个项目中实现该功能的代码——很不错的、紧凑的Java类，可以把经纬度组转换成x、y坐标。我们需要做的就是向Java库传递数据文件中的经纬度元组，Java库就会返回x、y坐标。然后，我们把这些坐标值传给Processing的绘图API来定位《纽约时报》Web站点和手机站点的每个用户的位置。

基础层地图

创建基础层地图所需的时间会远远超过你的想象。首先，我们需要对美国和世界做出准确的表示。经过大量的数据探索后，我们最终使用加州大学洛杉矶分校的CENS组数据集，它描绘了世界上每座城市的经纬度坐标。

在使用该数据集的初始阶段，每当程序启动时，直接在Processing集成环境中进行渲染，但这个渲染花费的时间比我们期望的要多很多；因为知道该数据不会变，最后，我们创建了一个JPEG地图，向背景地图中加载一个非常小的文件。这种方式给我们节省了好几分钟的渲染时间（当解析大数据集时，这部分工作所需的时间会更长）和处理能力，并且成为所有后续的数据输出和视频的背景。

刚刚处理的数据哪去了

有了纬经度投影代码和地图轮廓，我们开始在地图上描绘交通数据图。在可视化初期，我们使用不包含重大新闻的任意一天的数据（2009年2月15日）。这一天的Web站点和手机站点的流量/访问次数和平均值一致。

我们之前已经对数据进行过清洗、排序和添加地理位置编码，包含了时间戳、Web站点和手机站点上给定一天的用户每次查看/点击时所处的纬度/经度值。现在到了创建一个Processing应用程序的时刻了，它可以扫描Web站点和手机站点的日志

文件，对于用户的每次查看/点击，会在地图上描绘一个基于用户点击时所在位置而生成的点。

场景1，步骤1

Processing应用在绝大多数情况下由两部分组成：启动（setup）和循环绘制（draw）。在Processing应用的setup()函数中，你可以执行应用需要的任何工作，比如变量初始化、打开输入文件、字体加载等。循环绘制是Processing代码的根本。Processing应用中的draw()函数通常每秒钟会被调用30～60次（这是时间帧速率）。

我们第一次尝试的代码尽管存在一些问题，但能够生成一些可以在屏幕上观看的画面。可以多次运行该应用程序，查看图片中描绘的点，这些点表示《纽约时报》Web站点和手机站点一天的流量。随时间变化的流量的模式让人难以置信——画面看起来似乎是活生生的，闪烁的灯光散布在整个地球上，如图1所示。

这是伟大的第一步，但我们的代码和方法都需要做些修改。以下是需要改进的三个方面。

没有具体比例

首先，该可视化没有显示来自每个用户位置的Web站点和手机站点的流量的比例。比如，在一天的某个时刻，可能有很多Web站点和手机站点的用户是来自相同的地方，比如纽约，可以看到有非常高的流量。有时，可能有成千上万用户来自同一个地理位置。同样，假如是纽约！

在该应用程序的最初版本中，日志文件中出现的每个地理位置（一组经纬度值）在地图上都使用相同大小的点表示。为了能够表示比例，需要基于与某个位置关联的用户量来调整每个位置的可视化表示（地图上的蓝色和黄色点）。

图1　原始可视化显示了《纽约时报》Web站点和手机站点在全世界的流量——黄色圆圈表示Web站点的流量，蓝色圆圈表示手机站点的流量

其次，因为黄色（表示Web站点流量）和蓝色（表示手机站点流量）点大小相同，而我们（在绘制循环中）先画表示Web站点的点，再画表示手机站点的点，当两种点击类型位于同一个地理位置时，蓝色点会覆盖黄色点。这对可视化而言不是一个好的选择。

没有考虑时间

在可视化的第一阶段，我们没有考虑人们在Web站点或手机站点上每次访问或页面查看所花费的时间，只是简单地在地图上为每次访问画了一个点，在可视化的整个过程中都不再管它了。这样，就没有人会注意到在某些大城市《纽约时报》有持续较大的流量，而在一些小的偏远地区我们可能一天只有几次查看，这种表示方式会使我们错误地认为这些地区整天都有流量。

我们需要解决这个问题，并结合比例表示问题，也就是说，我们需要提出一种新的方法，可以精确地表示从任何一个位置有多少人访问该网站，以及他们在某篇文章上停留了多长时间，或者在整个网站上停留的时间。

定时拍摄

最后，我们选择将整天的数据流量创建成为一个定时拍摄视频，从而使我们能够在整个《纽约时报》公司内共享该可视化。为了解决这个问题，我们决定使用Processing的一个内置的视频库，它能够将循环绘制生成的时间帧保存到视频文件中，进而创建出很清晰的电影形式的输出。

场景1，步骤2

在第一个版本代码基础之上，我们增加了通过Processing的MovieMaker库将可视化捕获并保存到一个文件中的功能。我们还增加了应用支持，能够使一对Web站点或手机站点的每次点击的可视化都能够体现该次访问的生命周期。平均来说，Web站点和手机站点这两个站点的一次访问时间是历时3～4分钟。因此，在迭代过程中，不再是在地图上画一个点并在后面整整24小时都不管它，我们尝试慢慢地每3分钟淡出消减一个点。当然，一个独立用户不是每3分钟对Web站

点或手机站点执行一次点击——日志文件中显示的很多点击都是来自同一批用户，或者是用了更长的时间浏览了网站的很多页面的用户。但是为了避免可视化的最初版本过于复杂，我们就笼统地认为每次对网站的访问都是“3分钟访问”。

对于这种简化的表示，我们需要保存一天内的每次查看/点击淡出3分钟以上的点。这意味着需要在内存中存储很多对象。对于每秒钟内Web站点和手机站点上的每次点击，我们都会在Processing应用程序中创建一个对象，它的任务是保存该点击的“生命周期”，也就是说，这个点需要在屏幕上停留多长时间（3分钟），使用这些对象来帮助我们在可视化的整个周期内对点实现淡出效果。

因此，再回过来看Processing的绘制循环。我们还是每秒钟从Web和手机站点的日志文件中读取数据，但对于每次单击，创建一个Hit（单击）对象，其初始生命周期设置为3分钟，初始不透明度是100%（这些值在迭代循环的每次绘制中不断减少）。读完日志数据后，遍历内存中Hit对象集合。对于每个Hit对象，重新描绘表示该单击的点，其透明度是基于该单击剩余的生命周期，在3分钟时间内把它淡出。当每个Hit对象达到生命周期时，把它从内存中删除，并从地图上删除相应点（即不再重新描绘它）。

因为每秒钟大约需要对400～500次点击进行可视化，这种方法意味着任何时刻都需要在内存中存储很多对象，来保存所有点击轨迹。

可视化的第二步

为了实现想要的可视化，除增加支持能够显示每个地理位置的流量比例，需要对应用程序进行优化，还需要重新思考如何收集数据。

重新回到比例问题

每秒钟显示每次点击并不能显示任何比例。在第一版的应用程序中，来自加拿大农村地区的少量的点击和来自纽约的成千上万的点击，其可视化权重是一样的。此外，从内存和处理器对可视化进行渲染的处理能力而言，每秒钟显示所有的点击代价太高。

想清楚后，我们认为答案是对每分钟每个地理位置的点击次数进行可视化，而不是每秒钟进行可视化。对于访问日志文件中的每分钟的数据，我们会累加每个地理位置的点击总数。这种方式使得可视化结果可以显示每个地理位置的流量比例，而且会极大地减少Processing应用程序的原始数据输入。但是，这种方式意味着我们需要改变数据处理和map/reduce作业。

进一步处理数据

之前用Python实现的map/reduce脚本，其目的是从原始访问日志中解析出我们需要的数据，并基于时间对数据进行排序，因此，需要做些修改。现在，该脚本需要对每分钟、每个地理位置（一组纬度/经度值）的所有点击进行计数，输出结果数据并根据访问时间进行排序。

从根本上说，map/reduce是一个编程模型，支持海量数据处理。其处理过程分成两个任务：mapping（映射）和reducing（规约）。Mapper通常是接收一些输入（在我们的例子中是日志文件），对数据做一些较小的处理，然后以键/值（key/value）对的方式输出数据。Reducer的任务是接收Mapper的输出结果数据，对数据进行归并或规约，通常生成较小的数据集。在我们的应用程序中，Mapper脚本读入原始的访问日志文件，对于每一行，以如下格式输出键/值对：

```
Timestamp of the access (in HH:MM format),latitude,longitude 1
```

在这个例子中，key是以逗号作为分隔符，包含了日志文件中每次点击的时间戳、纬度、经度，而value是1（表示一次点击计数值）。

然后，Reducer逐行读取Mapper的输出，保存每分钟每个地理位置的点击计数值。因此，它把Mapper输出的每个“key”存储到一个Python字典中，每次遇到Mapper的输出有相同的“key”，就把其在字典中的计数值增加1。Python字典看起来大概如下：

```
{
        "12:00,40.7308,-73.9970": 128,
        "12:00,37.7791,-122.4200": 33,
        "12:00,32.7781, -96.7954": 17,
        # cut off for brevity...
        "12:01,40.7308,-73.9970": 119,
        "12:01,37.7791,-122.4200": 45,
        "12:01,32.7781, -96.7954": 27,
        # ...
}
```

一旦Reducer读取了Mapper的所有的数据输入，它对数据进行排序（基于key），然后输出排序的结果。

新的数据格式

在原始访问数据上运行完新的map/reduce脚本后，我们得到了一组更准确的数据集。这个过程不仅减少了总的数据量（Web站点的访问数据，从3000万行左右减少到300万行），而且为我们生成了每个地理位置的计数值。现在，我们需要确定比例因子。以下是新的结果数据的样本——注意时间戳、纬度、经度和（每分钟的）点击计数值。

```
12:00,039.948,-074.905,128
12:00,039.949,-082.057,1
12:00,039.951,-105.045,3
12:00,039.952,-074.995,1
12:00,039.952,-075.164,398
12:00,039.960,-075.270,1
12:00,039.963,-076.728,4
12:00,039.970,-075.832,2
12:00,039.970,-086.160,4
12:00,039.975,-075.048,23
```

可视化比例和其他可视化优化

有了新形式的数据，我们不再是每秒钟为每次点击画一个点，而是可以每分钟为每个地理位置的点击数值画一个圆圈，并根据点击数计算圆圈大小。这种方式可以生成期望的比例显示，使得可视化的读者可以轻松地区分来自加拿大农村和纽约市的不同的流量差别。

这种方式也极大地减少了应用程序需要的内存量。我们还是需要在内存中保存Web站点和手机站点的所有点击（这样我们才能消隐去时间超过3分钟的点击），但是因为我们现在保存的是每分钟每个地理位置的点击数，极大地减少了需要的Hit对象数量。对于任一分钟，来自全世界的流量通常包含2000～3500个不同的地理位置。每个位置的Hit对象必须存储在内存中；每个Hit对象生命期是3分钟，因此对于任一时刻，内存中可能有6000～12 000个对象——数量还是很多，但是已经远远小于前一版本的对象数量。

现在，需要更新Processing应用程序，从而可以实时保存每个位置在任一时刻的点击数，而且圆圈大小比例可以根据点击数调整。

使定时拍摄能够正常工作

对Processing应用程序进行升级使其能够处理新的数据格式和方法，在此之后，我们创建了一个完整的历时24小时的定时拍摄视频。我们新的代码每次能够正常运行几个小时，不存在之前遇到的内存和整体机器延时，现在是生成完整的定时拍摄视频的时候了。不再像第一次那样尝试在地图上为历时24小时定时拍摄渲染Web站点和手机站点数据，我们只使用手机站点的数据（其数据量大约是Web站点数据量的10%）。这样，我们就可以比同时渲染Web站点和手机站点数据更快地查看到结果或者发现可能存在的问题。

由于不确定应该对24小时的定时拍摄进行多大程度的收缩，我们决定测试一下，采用10分钟。该项目最激动人心的时刻之一是当我们首次尝试渲染24小时的手机站点数据时，点击Processing的运行（Run）按钮那一刻。把数据在一台MacBook Pro机上渲染成10分钟的定时拍摄视频花了约2个小时。结果生成了！

大家互相击拳祝贺后，开始观看视频。看了大约两分钟，我们意识到视频时间太长了——感觉视频太慢了！开始重新装载数据，创建一个历时接近1.5分钟的视频。经过几次尝试以及对代码和帧速率的调整，我们生成了新的视频。对较小规模的手机站点数据集进行渲染可以正常工作后，我们开始在Web站点和手机站点的混合数据集上尝试。由于数据量比之前大得多，渲染花费的时间也长很多——之前是2个小时，这次渲染花了24～36个小时，这取决于其所用的机器的性能。

半自动化

最后，我们希望能够对整个过程实现自动化，这样程序接收到输入命令后，可以执行任何一天的定时拍摄渲染。该过程现在是半自动化的，我们可以很容易为同一天渲染多个定时拍摄的视频。举个例子，我们可以针对以下任何一种情况进行渲染：

- 世界地图的Web站点和手机站点的数据。
- 美国地图的Web站点和手机站点的数据。
- 世界地图和美国地图的Web站点的数据。
- 世界地图和美国地图的手机站点数据。

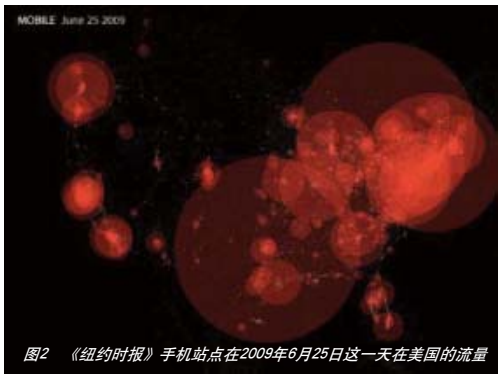


图2　《纽约时报》手机站点在2009年6月25日这一天在美国的流量

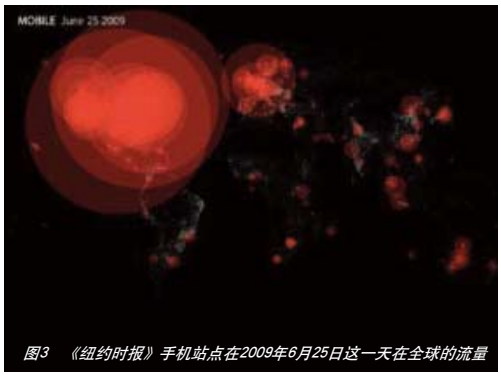


图3　《纽约时报》手机站点在2009年6月25日这一天在全球的流量

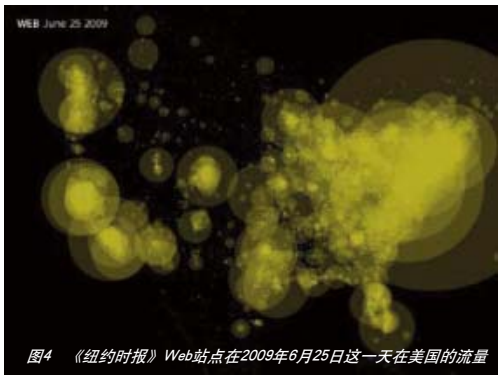


图4　《纽约时报》Web站点在2009年6月25日这一天在美国的流量

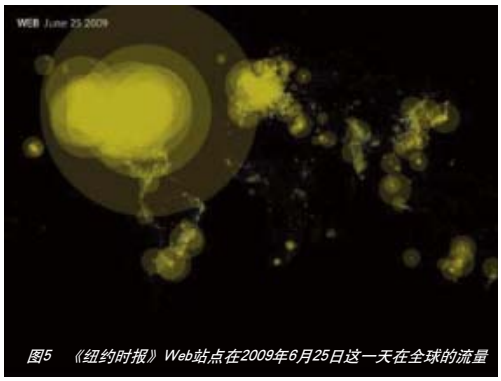


图5　《纽约时报》Web站点在2009年6月25日这一天在全球的流量

每种类型的数据需要花多长时间渲染？这取决于日期以及那一天是否是重大新闻日（即是否有很大流量）。

渲染定时拍摄视频的数据计算

在Processing应用程序内，我们每秒钟捕获15帧的视频。对于每一帧，在屏幕上绘制了1分钟的日志量。对于24小时的数据量，需要捕获1440分钟的数据。把每15分钟的数据渲染成时间长度为一秒的视频，则1440分钟的数据会生成96秒钟的视频（约1.5分钟）。

生成的视频有什么用

在纽约时报大厦28层的走廊上挂着10台监视器，播放着我们所做的一些可视化视频，包括这些流量图。其中有6台监视器自动播放本章介绍的定时拍摄视频；其他4台屏幕上显示的是《纽约时报》Web站点和手机站点当天全部流量的快照（美国和全球）。我们开始在公司内分享这些视频，并且探索更多的可视化来查看一天内可以发现哪些模式。我们还观察“重大新闻日”和“平常日”中，用户使用模式的差异。

结束语

我们从目前创建的可视化中观察到了一些有趣的模式，绝大多数如图2~图5所示。

第一个模式是手机站点的流量在美国约早上5点或6点开始暴涨，该时段人们醒来开始去上班（尤其是在东海岸）。在约8点半或9点人们到达办公室前，手机站点流量一直很大；而当人们到达办公室时，Web站点流量开始第一次大增。Web站点的流量在一整天都很大（尤其是午饭时间），下午稍有点下降，很可能是人们在下班路上，而这时手机站点的流量又开始增加。这个观察和我们开始研究前的预期相同，但是该可视化进一步证实了我们的猜想。

另一个有趣的模式是Web和手机站点的国际流量都很大，非洲、中国、印度和日本某些地区的手机站点流量也很大。

本文节选自机械工业出版社华章公司《数据可视化之美》一书。该书介绍了数据可视化的方法和思想，通过描述分析很多实例，带领读者深入洞察数据可视化之美。特此感谢华章公司与Michael Young和Nick Bilton先生授权。

好钢用在刀刃上

豆瓣网开发经验分享

文 / 解彦博

优秀的员工，是豆瓣的“好钢”；工作效率和团队热情，就是豆瓣一直在锤炼的“刀刃”。

在国内的互联网企业中，豆瓣是一家非典型的公司，始终保持着自己独特的企业文化。我们率先使用与众不同的开发语言，也一直坚持走自己的发展道路，不为外界所动。在这些年的成长过程中，豆瓣积累了一些开发过程方面的经验。通过与大家分享这些经验，希望能抛砖引玉，一起促进、共同成长。当然，没有能适用于任何公司的万能方法，最终还是需要大家去尝试、探索，找到真正符合自己情况的做法。

在中文维基百科上，“开发过程”是这样定义的：“软件开发是根据用户要求建造出软件的一个产品开发的过程”。豆瓣是互联网公司，开发团队面对的客户是公司内部的产品经理。众所周知，互联网公司的一大特色，就是：快。市场变化快，需求变化快，开发过程也需要快。而在豆瓣的发展过程中，与豆瓣网站上各种丰富的产品相对应的，总是相对较少的开发人员。因此，豆瓣的开发过程，是“利用有限的人力和资源，根据网站需要快速建造出合格产品的开发过程”。这迫使豆瓣的开发团队保持了一种类似创业团队的文化：追求高效工作方式，保持像创业团队一样的工作热情。

优秀的员工，是豆瓣的“好钢”，工作效率和团队热情，就是豆瓣一直在锤炼的“刀刃”，是我们努力和提高的目标。

效率

开发语言选择用Python

说到效率，豆瓣本身非典型特性的一个体现，就是在最开始时，选择了Python作为开发工具。Python这个词，原意是大蟒蛇、巨蟒。作为一门高级语言，它的语法是面向对象式编程与函数式编程混合的，很符合开发人员胃口，哪种方式用起来更简便，就可以选择用哪种。另外它也是脚本语言，源代码文件可以直接执行，不需编译，很节省开发时间。但Python也有一些弊端：由于是动态化高级语言，会出现只有在执行时才能发现的问题。比如由于豆瓣的代码规模越来越大，模块间的关系特别复杂，循环依赖（Circular Dependency）出现得越来越频繁，而且往往涉及3~4个模块，很难事先发现。这个问题其实是由于模块的组织方式不够好，我们目前正在做一些工作来解决它。总体来说，使用Python语言开发豆瓣，获得的利益远远大于带来的麻烦。

下面我举两个例子来说明Python在豆瓣的使用。

第一个示例是交互式的控制台。Python的一大优势是可以交互式执行，豆瓣借用了这个特性，在控制台启动时，把所有豆瓣网站代码都已经装载进来，就可以直接调用这些已有的代码。很多临时性或一次性的工作，都可以用这个工具迅速完成。如图1所示，是通过封装

```
xyb@ori /var/shire $ ./shell.py
Python 2.6.4 (r264:75706, Mar 15 2010, 05:04:29)
Type "copyright", "credits" or "license" for more information.

IPython 0.10 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object'. ?object also works, ?? prints more.

In [1]: mc.set('email_index', 1000)
Out[1]: 1

In [2]: mc.get('email_index')
Out[2]: 1000
```

图1 交互式控制台

好的缓存接口，向email_inbox变量存入一个数值，然后再取出来。这个工具是做一些简单的调试、运维工作的利器，一些只需要执行几行代码的工作，都可以通过它来完成。比如线上的一些简单的控制开关，都可以通过这种方式快速切换，改变线上功能或者展示界面。

Python其实是与Java同时出现的，在这些年的发展中，积累了大量成熟的第三方代码。可以轻松地在这些代码的基础上，写出简洁但能实现复杂功能的代码，从而极大地降低了开发成本，减少了开发时间，提高了工作效率。

第二个示例是豆瓣的一段代码，利用PIL实现图片的蒙板透明和打水印两个功能，如图2所示。

```
from PIL import Image, ImageFile
from PIL.ImageEnhance import Brightness
from PIL.ImageDraw import Draw

def make_opacity(im, opacity):
    if not (opacity >= 0 and opacity <= 1):
        return im

    im = im.mode == 'RGBA' and im.copy() or im.convert('RGBA')
    alpha = im.split()[3]
    alpha = Brightness(alpha).enhance(opacity)
    im.putalpha(alpha)
    return im

def watermark(im, markim, coordinate):
    layerim = Image.new('RGBA', im.size)
    layerim.paste(markim, coordinate)
    return Image.composite(layerim, im, layerim)
```

图2　使用PIL实现图片透明化和图片打水印功能

在图2的示例中，使用了PIL模块的两个函数，分别实现了图片透明化和图片打水印功能。PIL是一个开源的Python语言处理图像模块，历史悠久，久经考验。Python网站的图像处理就是在这个模块的基础上开发的。

灵活的工作流程

开发是一个复杂的过程，会涉及很多流程。僵化的流程会降低效率，一个团队不会在创业伊始就用复杂的流程束缚自己的手脚。同样的，虽然随着团队的增加，豆瓣需要明确一些工作的流程，但在不断做过程改进的同时，我们仍然保留了使用更原始但更快捷、高效的流程的可能。

早期，每位开发工程师，都独立承担开发、上线的全过程。这一方面是由于开发环境、运行环境、上线方式都相对简单；另一方面也很重要的是，工程师应该为自己编写的代码质量负责。他应该有能力、同时也有责任为自己的代码的整个生命周期负责，从代码的设计、编写、测试、上线、维护直到重构和消失。不过随着代码复杂度的不断增加，每次上线需要做的工作越来越多，这逐渐成为开发工程师的负担。现在，上线过程已经使用自动化工具执行，并由专人负责，以便把工程师解放出来，做更有价值的工作。但作为一个普遍规律，必须认识到所有事情都可能出问题。当服务器足够多时，每天都会有硬盘损坏；当上线的代码足够多时，总是会有重大Bug被发布到线上。这时就需要有紧急上线机制，在这种机制下，平常开发、上线流程中的代码复审、测试等环节都可以取消。这种紧急上线，可以在几分钟之内完成，让网站和用户受到的损失最小化。需要强调的是，上线流程专人化最重要的目的是解放开发工程师的生产力，减小上线风险，而非限制工程师的权限。在未来，当上线流程的工具化、自动化以及其他一些条件成熟后，上线工作仍将由每个团队、每位开发工程师负责。

日常工具

所谓“工欲善其事，必先利其器”，豆瓣有很多类似上线工具的脚本、辅助系统等，这些工具把人从烦琐的日常事务中解放了出来，是提高工作效率必不可少的手段。比如前面提到的shell.py，可以方便调试和维护。

此外，豆瓣还有各种代码测试、检查工具，帮助提前发现程序中的问题。比如豆瓣使用pylint做代码的静态扫描。pylint是一个开源的Python静态分析工具，它本身支持自定义扩展，我们按照豆瓣的需要定制了一些检查功能，比如扫描跨站脚本引用（XSS）、SQL脚本注入（SQL Inject）的检查等。这个工具既可以自己运行，也会作为持续集成的一部分自动执行。通过自动扫描、检查代码，帮助发现隐患，这个工具让程序员的开发更有效率。

```
lus/survey/radio/survey_ui.ptl:34 [W9802] Maybe an XSS hole, use js_quote
docs/nav.html:174 [W9803] Maybe an XSS hole, use form var in template
docs/utils.html:572 [W9802] Maybe an XSS hole, use js_quote
docs/utils.html:585 [W9802] Maybe an XSS hole, use js_quote
```

图3　Python静态分析工具pylint

是程序总会有Bug。Python语言的程序出错时，会留下traceback，其中包含了出错原因、涉及的代码、调用过程等。由于所有错误发生

在众多不同的Web服务器上，不利于工程师收集、调试，所以我们开发了这样一个工具，抓取所有错误日志，并把出错现场的各种详细信息，比如变量中的值等，全都保留下来。工程师就可以从一个地方快速浏览错误日志，发现Bug。有了这个工具，大部分时间工程师不用复现Bug，就可以直接定位代码并修复。通过持续改进这个工具，使程序员的调试更有效率。

图4 错误日志分类图

对开发团队来说，不只是人做事要有效率，运行的程序同样需要高效。豆瓣的服务器并不多，上面程序的执行效率对我们来说意义重大，高效的程序可以帮我们省钱。我们很早就做了一个工具，帮助直接在线上检查程序的执行效率。使用它，可以直接看到要生成一个页面，底层执行了那些代码，对数据库做过那些请求，花费多少时间等；依靠它，我们把平均访问时间保持在一个比较小的范围之内。通过在线性能优化工具，可以很方便地发现热点，从而提高程序员做性能调优的效率。

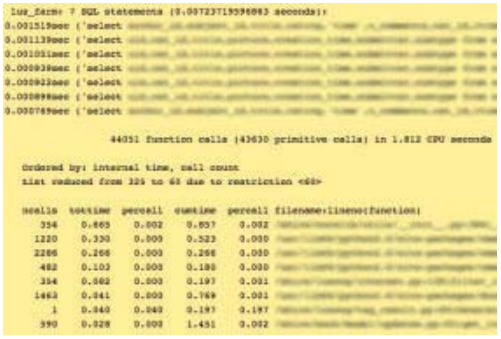


图5 线上检查程序执行效率的工具

自动化

虽然有了很多类似上面的工具，但这些工具还是需要人去定期执行、检查执行结果，仍然会占用工程师的时间和精力。通过把工具自动化，在不需要人员介入的前提下，仍然发挥工具的作用，是对解放工程师非常有帮助的。通过持续集成系统，豆瓣已经将单元测试、Web测试、静态分析等多种开发辅助工具，实现了自动化执行。持续集成系统只有在发现问题时，才会把报告通过邮件发送给工程师，这样做到对工程师最少的打扰：没有消息就是好消息。

图6 Web测试工具

选择空间

前面提到的都是各种工具和软件，但对实现高效率最核心、对保持开发团队创造力最重要的，是给团队提供自由灵活的技术选择空间。很多已经形成规模的公司和团队，局限在已有的框架中，往往容易失去创造力和活力，工作效率也越来越低。反观创业团队的高效率：在做前人没有做过的事情时，没有任何束缚，才能选择最好的方案，既快速又高效地完成想做的工作。我们希望通过提供更灵活的选择空间，让工程师发挥新技术的优势，避免架构僵化、衰老，从而保持团队的创造力以及对于新技术的敏锐性，使团队不断获得新的生命力。通过公司架构师等资深同事的讨论和参与，可以尽量减少选择新技术时引入的风险。

我们在使用原有技术做网站开发的同时，还在不断尝试新技术。比如，我们使用的Web开发框架，豆瓣网基于quixote，公司内也有团队

用Django、pylons等；虽然公司的主流开发语言是Python，也有同事和项目是基于C、R，甚至Google的新锐语言Go。豆瓣也不断开发自己的基础设施，比如已经开源的BeansDB，这些可以让工程师在开发时无须关注自己不感兴趣的底层细节，多关注业务逻辑，使开发更加方便，提高了开发的效率。

工作热情

不过，即使有了很多高效率的工具，如果团队死气沉沉，每个人对工作没有兴趣，那么这个公司也还是很失败。作为还在不断成长的公司，我们很注重团队的工作热情。保持团队的热情，首先还是得从人做起。

优秀员工

人是团队最基本的元素。Facebook前工程总监黄易山说道：“永远将招聘做为你的第一要务”（注：原文参见《程序员》2011年3月刊）。豆瓣也一样对招聘非常重视。优秀的员工、团队成就了豆瓣，也造就了今天的豆瓣文化。通过观察已有的这些优秀员工的品质，作为豆瓣招聘的指导原则。我们从能力、与公司文化是否契合等各方面考察应聘者。与招聘职位相关的团队成员、会有合作的同事，也会参与招聘。他们是否愿意接纳应聘者作为自己的工作伙伴，是非常重要的指标。

办公环境

工作环境也是一个很重要的因素。豆瓣的办公室开放、通透，办公室的布局鼓励大家互相交流。

在软件环境上，我们尽量简化各种杂务的处理方式，节约工程师的精力和时间。比如一键式初始化开发环境，以及登录即可使用的开发虚拟机。我们希望工程师开始一项工作时满怀的热情，不会被各种杂务消磨掉。

减少限制

为了提高效率，我们在开发方面一直尽量减少限制，让工程师能有更自由的选择空间。为了提高和保持团队热情，在行政方面，我们也同样避免行政限制，取而代之的是给一些建议。比如，不把代码的测试覆盖率作为一项强制指标。行政强制需要谨慎，应该作为最后的手段，三思而行。

完成自己想做的事情是快乐的，工作热情来源于此。应该充分发掘每个人的兴趣，找到喜欢做、适合做的事情。比如有人爱研究技术，把新技术应用到现有框架中的工作，会让他产生很高的热情；有人对代码有洁癖，可以考虑让他做代码重构、改进的工作。通过发现和鼓励每个人热情的来源，可以让团队保持良好的工作热情。

发展成长

另外，通过各种技术交流、培训等，让每个员工都能不断获得成长，也是保持工作热情的一个重要方面。在豆瓣内部有各种活动，比如团队内部技术交流、团队间合作及交流、公司内部培训、请公司外的高手来讲座、报销购书费用鼓励自学、Happy Day活动在游戏中学习知识等。

所有的开发活动，最终都需要由人来完成。人是整个过程中最宝贵的。豆瓣保持着类似创业团队的文化，但我们不是要求大家天天加班，不是用更多的工作时间换取更多的业绩，而是通过营造更好的环境，使大家能更好、更高效地发挥自己的能力，取得更多的劳动成果。当团队中所有人都拥有相同的目标，能充满热情、高效率地开发，这个开发团队就能发挥出与创业团队一样巨大的潜能。希望每个公司、每个开发团队都能把好钢用在刀刃上，实现自己的价值。

解彦博

豆瓣网QA主管。豆瓣早期员工，从事多年互联网开发工作，热衷于任何可以让开发变得更舒畅的方式和方法。

责任编辑：董世晓（dongsx@csdn.net）

产品管理的前世今生——昨天

文 / 张智渊

作为“产品管理的前世今生”系列文章首篇，本文介绍了产品管理体系所依赖的理论基础、所属的学科范畴、经历的阶段以及一些常识性的概念错误。

说到产品管理体系的诞生，问10个产品管理者，会有9个告诉你宝洁的故事。确实如此，国内几乎所有的产品管理者，不分行业，都无一例外地知道产品管理者诞生于宝洁公司，并以此作为工作的标杆而津津乐道。

但如果你要继续追问：“是什么原因促使宝洁要设立产品管理者”，就会有相当多的人说不出来了，即使知道原因的也只能告诉你：“是为了应付日益加剧的市场竞争”。这难道真的是产品管理体系诞生的根本原因吗?

每门管理学科的产生，必然和现实的社会环境、市场环境、企业环境有关。从宝洁采用产品管理体系到现在已经有80年的历史了，并且越来越为企业认可和采用，从这个角度就说明，产品管理体系的诞生和发展并不是因宝洁而起，而是因为当时大的市场环境要求有一种新的有利于企业竞争的架构出现。幸运的是，宝洁推出的产品管理体系开始适应新的市场环境，成为翘楚。可以这么说，是市场成就了宝洁的产品管理体系并使之发扬，而不是宝洁成就了产品管理体系。

同样，一门管理学科必定是随着市场环境的变化而发展的，80年前的宝洁模式是否适用于目前国内的市场现状，是否适合具体的企业环境，这是每一个产品管理者都需要考虑清楚的。

因此，本系列文章将从三个部分来介绍产品管理体系从诞生、现状以及将来发展的问题，旨在通过对产品管理体系进行一个全面的剖析，重点介绍产品管理体系所依赖的理论基础、衍生过程、现状分析以及未来走向。

本篇将重点介绍产品管理体系所依赖的理论基础，所属的学科范畴，经历的阶段以及一些常识性的概念错误。建议所有产品管理者参考。

关键词：Marketing

首先要明确一个产品管理体系最基础、最核心的概念：Marketing。

Marketing，中文通常翻译为“市场营销”，那到底什么是市场营销呢？是否就是大家通常看到的企业市场部门所做的那些工作呢?

到目前为止，国际上也没有一个明确的定义，通常分为“窄派定义”和“宽派定义”，但现在被大部分人所认可的基本都是“宽派定义”，而在宽派定义中，同样也有不同的说法，但最常见的定义是以下的说法：

市场营销是个人和机构通过预测、刺激、提供方便、协调生产与消费以满足顾客和社会公众对产品、服务及其他供应的需求的整体经济活动。

看到这个定义，是不是感觉有些熟悉，是不是和产品管理者的根本工作定位有些一致呢?

再来看一下产品管理者的根本工作定位：

产品管理就是依据市场需求，通过有效组织企业资源来满足用户需求而提供产品或者服务的过程。

Marketing和PMS两个概念的比较见表1。

那么，PMS和Marketing是什么关系呢？难道说，PMS就是Marketing吗?

表1 Marketing和PMS两个概念的比较

	Marketing	PMS
执行者	个人和机构	产品管理者
目标对象	顾客和社会公众	目标用户
工作过程	预测、刺激、提供、生产、交易	预测市场、分析需求、规划产品、生产协调、市场交换等产品生产全部过程
工作目标	满足需求	满足目标用户需求
工作成果	产品或者服务	产品或者服务

Marketing的发展过程

在回答这个问题前，必须要重新回顾一下Marketing的发展过程，否则将无法更好地理解两者之间的关系和区别。

19世纪60年代~20世纪初，西方第二次工业革命的发展导致工业化生产成为社会主要生产形式，大量的工业品被生产出来，即使这样，整个市场环境也处于“卖方市场”，因此那个阶段是处于以“生产”为中心的阶段，如图1所示。

图1 以“生产”为中心的阶段

在此阶段中，企业只要通过提高产量，降低成本，就可以获得巨额利润。在这个阶段中，企业的焦点是在考虑如何“生产”的越来越多，而不是如何“卖”出去，因为只要生产了，就一定能卖出去。因此，那个阶段的企业只有生产部门而很少有销售部门，即使有，其作用也非常小。

正因为如此，生产企业从不考虑用户需要什么，而通常是按照企业自己的想法来设计和生产产品，要么是主观地认为“消费者喜欢的是那些随处可以买到、价格低廉的产品”，要么是认为“消费者喜欢的是质量最优、性能最好和特点最多的产品”，这种观念直接造成了“营销近视病”，即过分重视产品本身而不是用户需求。

但随着市场的不断发展、技术的不断革新以及产品成本的不断降低，越来越多的个人和机构开始投入到工业中，进入者的增多造成了市场竞争的加剧，产品开始出现滞销，这就进入了第二个阶段：以“销售”为中心的阶段。该阶段的形成时间大概是20世纪30年代~40年代。

在此阶段中，竞争的加剧造成每家企业的产品库存都开始增加，企业发现“货不那么好销”了，因此，它们就开始花大量的人力、物力开始做广告、做宣传、建设自己的推销队伍，通过“高压硬卖”的方式来把自己的产品销售给用户，而不问是否真正适合用户，在这种情况下，伴随销货的必然是“不择手段，夸大其词”的短视做法。

因此，可以看出，在这个阶段，销货已经代替生产成为企业第一工作任务。如图2所示。

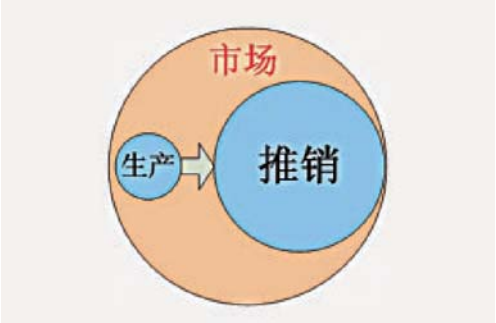


图2 以“销售”为中心的阶段

但是好景不长，毕竟用户不是傻子，因此，这个阶段持续了不到20年时间，市场开始进入到第三个阶段：以“消费者”为中心的阶段。这个阶段始于20世纪50年代。如图3所示。

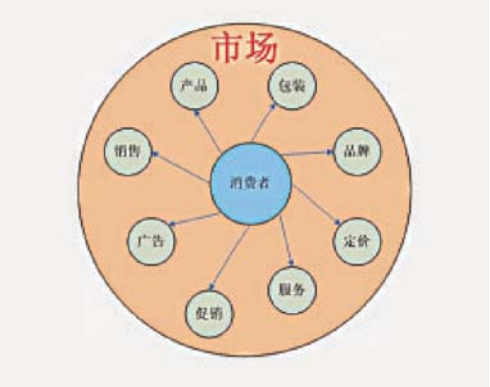


图3 以“消费者”为中心的阶段

在这个阶段，企业发现，要保持获得利润，使企业能够长久发展，提高销量，单靠扩大生产和推销，已经不能适应新的市场了，只有转变企业角色，尊重用户，了解用户的真正需要才可以。因此，这个时候，企业开始更多地倾听用户真正需要什么，开始考虑用户的需求，开始对用户的需求进行细分，开始有针对性地开发产品，开始采用比较“软”的方法来销售产品。

可以这么说，从20世纪50年代开始，Marketing这门学科才真正开始形成和快速发展开来，而随着这门学科的发展，许多企业发现现有的产品生产架构不能适应变化了的市场问题，当时企业面临的问题主要以下几个方面。

- 日益细分的市场和用户需求。
- 为满足这些需求而日益增加的产品种类。
- 如何才能把合适的商品销给目标用户。
- 不能满足于眼前，而要着眼于长远。
- 如何才能稳固现有用户，并吸引潜在客户。

而在当时，企业的产品生产架构是以“企业生产”为出发点的，如图4所示。

图4 以“生产”为出发点的企业产品生产架构

而这种结构已经无法解决现实的问题，那应该如何解决这个问题呢?

接下来的事情，大家就都知道了，就是宝洁推出了“以用户”为出发点的产品管理体系，并且在日后的发展中逐步完善，获得了全球大部分企业的认可。

解读PMS与Marketing的关系

现在可以回答“PMS和Marketing之间是什么关系”的问题了。简而言之如下。

- PMS是属于Marketing范畴的，也就是基于“市场营销”学科的。
- PMS是Marketing现阶段最典型和最有效的产品生产体系架构。

从词形看，Marketing是一个“动名词”，因此PMS所属于的市场营销应该是一个“活动过程”，而不是一个“组织形式”。而这个活动过程包含的具体阶段有“评估用户需求、整合企业资源、满足用户需求、实现企业目标”，如图5所示。

图5中具体的内容包括以下方面。

- 评估用户需求，包括市场分析、市场预测、计划制订、产品策略、产品规划。
- 整合企业资源，包括资源调配、产品生产、渠道建设、产品销售、售后服务。
- 满足用户需求，包括产品交易、市场反馈、计划修正。
- 实现企业目标，包括利润实现、价值实现。

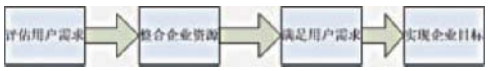


图5 PMS包括的具体阶段

可以看出，Marketing的过程是包含了企业产品生成整个过程的，正如美国的Burruoghs Corpration的董事长所说：“任何公司无非就是一个市场营销组织而已。”

而要实现这种形式必然需要一种架构体系来担当，这样，宝洁推出的产品管理体系就成为了Marketing的组织架构形式，这就是PMS诞生的理论基础。

再看第二点，为什么说PMS是Marketing现阶段最典型和最有效的产品生产架构呢?

在前面说到了，进入到第三阶段后，企业普遍面临的问题是：“需求的细分”、“产品的众多”、“资源的有限”、“长期的计划”、“持久的利润”、“客户的增加”。

这些问题中不止一对矛盾，而要解决这些矛盾，我相信，在当时，不止一家企业尝试过不止一种的组织架构形式，但无论采用哪种组织形式，最根本的目标就是“把有限的资源投入到最能长期盈利的产品上去”。

如果企业只有一个产品，就比较简单了，公司总经理就可以来负责产品的市场营销活动过程。但那个时候几乎所有的企业都有“多产品、多品牌、多市场”的情况，而一个公司只有一个总经理，无论从精力和能力上来考虑，都根本无法照顾到所有的产品。因此，就需要从公司内抽调专门的人来负责这些产品，替代总经理来控制某一个产品或者某一类产品的市场营销活动，这些人，就被称为“产品管理者”。

这就是众所周知的宝洁要设立产品管理者的直接原因。

因此，可以这么说，历史选择宝洁是偶然的，即使没有宝洁，也会有其他公司被推到前台，而市场新问题的出现才是产品管理体系诞生的必然。

对几个疑问的解答

通过分析产品管理体系诞生的历史原因和市场背景，其实可以解答许多产品管理者朋友的几个疑问。

■ 产品管理者是偏市场还是偏技术?

造成出现这个问题的根本原因就是没有理解“Marketing”的意义，不要认为看见凡是含有“Market”的词汇就一定是局限于市场的。刚才提到，Marketing是一个活动过程，包含的方面有很多，其中还包括“产品生产过程”，但在好多人的概念中，产品生产怎么可能是属于Marketing范畴的呢?

无论是开发、生产还是销售，都属于Marketing范畴，产品管理是Marketing组织形式，产品管理者是产品管理的直接负责人，把他定位到市场或者是技术，都是概念混乱造成的。

■ 产品管理者到底是一个什么样的定位?

许多人会说“产品管理者是没有权力的小CEO”，这只是一种形象的说法。通过这篇文章，就可以看出，产品管理者到底是做什么的，说白了，就是在产品管理体系下，把自己负责的产品做好、做强、做长。

具体的工作包括两个方面：产品战略和产品战术。

其实在国外，公司制订产品年度计划，都是会由产品管理者来参与并且主导和执行的，反而产品战术上则做得不多。

但国内的实际情况又是什么样呢?

■ 国内有真正的产品管理者吗?

从目前的情况来看，没有。更多的只是做了真正产品管理者的部分工作，称之为“产品规划经理”、“产品市场经理”更准确。

真正的产品管理者是做什么的，看看Marketing的活动过程即可知道，但大家又做了多少呢?

在美国，为了更明确地定位产品管理者，现在又有了一个更形象的叫法——“PMM”，即“Product Marketing Manager”，如果你翻译成“产品市场经理”就错了，应该是“产品市场营销经理”。

这个称谓有两个好处：一方面可以区分同样缩写是“PM”的项目经理和产品管理者；另一方面更形象地说明了产品管理者工作的内容——如果只是Product，就让人容易理解成产品管理者只是做一些产品规划和产品定义的工作，而一旦加上“Marketing”，就更加明确了工作的内容是在产品的“整个活动过程”而不只是在于“评估用户需求”过程。

■ 产品管理者应该在哪些方面加强呢?

既然产品管理者其实就是执行Marketing活动过程，那在这个活动过程中，所涉及到的知识就有许多了，既有市场层面的，又有技术层面的。这些知识作为一个真正的产品经理是都需要了解的，“产品经理是杂家”，这句话没错，但是没有指明范围，知道了产品管理的理论基础，就应该知道在哪些方面加强了。

这篇文章大篇幅地介绍了Marketing的内容，这是非常有必要的，因为PMS不是凭空出现的，它的出现有其历史原因和理论基础。只有了解了历史原因，知道了理论基础和出现的市场条件，对于个人来说，才能更好地把握现在自身工作的方向，发现现实的问题，进而进行修正和完善。对于企业来说，才能更好地来定位产品管理者，定义这个职位，并赋予相应的责权利，而不要再出现把产品管理者定义到“市场部”或者“研发部”这种可笑的情况。

总结

在本文中，大家需要注意的几个方面。

■ 真正理解“Marketing”的含义。

■ Marketing是PMS的生存土壤和理论基础。

■ PMS是Marketing的必然产物。

■ PMS是随着Marketing的发展而发展的。

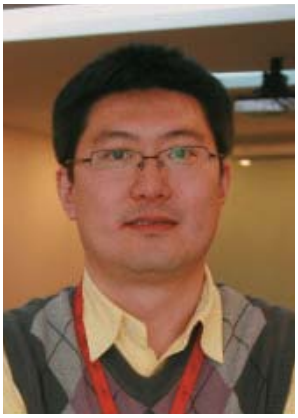
■ 两者是“源之水、本之木”的关系。

张智渊

UCPM中国产品经理联盟发起人，长期关注中国产品管理体系的建设和发展，曾在国内多家知名企业从事产品管理工作，具有丰富的一线产品管理经验和扎实的理论基础。

责任编辑：董世晓（dongsx@csdn.net）

如何管理“问题员工”



崔立元
奥博杰天软件有限公司交付运营总监

只有需要被了解、关心和关注的员工

在我的概念中，不存在“问题员工”，只有需要被了解、关心和关注的员工。如果非要找出“问题员工”，那么我认为所有员工都有机会是“问题员工”。当然在工作中也会遇到某些员工工作积极性不高、与他人沟通交流不畅、工作中缺乏责任心等现象，而这些恰恰就是我们需要关心和关注的。

作为管理者来讲，你真的懂你的员工吗？员工有自己的职业规划，有自己感兴趣的东西，有自己的特长和自己的个性，有自己的生活和工作方式……只有了解了这些，才能真正去关心员工，帮助员工解决问题（issue）。在奥博杰天，每位员工都会定期与项目和团队的管理者进行一对一的聊天，他们都有机会把自己的想法和建议提出来，而管理者也会将工作中发现的一些问题（issue）与员工进行充分交流。一对一的聊天只是日常工作中沟通交流的一部分，我们鼓励员工把自己的想法和建议通过各种方式提出来，而管理者也会在工作和工作之余充分与员工接触和交流。为了解决问题，管理者通常会与员工一起制订一个1~2个月的计划，并且在计划执行的过程中不断地给予反馈，特别是当员工在某个方面表现出色时，我们一定会积极给予正面的反馈，激励他/她继续努力。

作为管理者来讲，你的员工都在适合他们的位置上吗？人无完人，怎样用好员工，特别是一些“奇才”和“怪才”，把他们的长处真正发挥出来？我们的核心文化之一就是灵活性（Flexibility）。对于一些有特长的人才，我们尽量根据他们的长处灵活地安排他们的工作，甚至创造新的职位给他们广阔的空间。一旦能够把自己的聪明才智用到擅长和感兴趣的工作上，他们就会给周围的同事甚至一个团队带来积极的影响。IT工程师转型成为开发人员，英语老师成为项目经理，QA工程师变成行业领域专家，这些都是真实发生在奥博杰天的事！

我们欢迎“不听话”的员工，欢迎员工提出自己的意见，甚至是与管理者相反的想法，特别鼓励有创新和创造性的想法与建议。如果说真的有“问题员工”的话，欢迎他们加入。

取其长处，容其短处

任何一家企业，无论是规模庞大的500强企业，还是成长中的中小型企业，都会存在“问题员工”。“问题员工”的出现，不仅会降低其自身的工作效率，而且会给周围的人带来一些负面影响，并且有可能影响到整个企业的运转效率。因此，我们历来都非常重视“问题员工”的管理。

在我看来，“问题员工”可以分为以下几种类型：第一种是所谓的“功高盖主”的员工，他们通常业绩非常好，所以往往不太重视、甚至不遵守公司的规定；第二种是那些有活力、有创意、有想法、标新立异的员工，他们创造出来的成果对提高整个公司的工作效率都会有很大帮助，但正是由于这些个性，使得他们不怎么重视公司现行的一些规章制度，甚至可以说是不屑一顾；第三种是那些非常苛

何剑

易到用车网人力资源总监

求的员工，他们爱挑毛病，在工作中表面看起来是追求完美，实际上往往会陷于一种吹毛求疵的境地，对同事、对合作部门的要求都非常高；第四种是那些喜欢推卸责任的员工，他们往往夸夸其谈、光说不干。

我们一直把人才作为最核心的价值，在招人、用人、育人、留人、成就人的各个环节中，都非常重视和尊重每一位员工的价值，这里面，既包括优秀员工，也包括“问题员工”。“问题员工”的管理，实质上是育人和留人环节中的员工管理。对于不同类型的“问题员工”，在进行管理时，我都遵循着一条共同的准则：“问题员工”不会自觉消失，必须采取有效的办法来解决，不武断地否定，认真发现其长处，适当容忍其短处，通过正面的引导，帮助他把短处转化成长处。

具体来说，对“问题员工”的管理，可从以下几方面来看。

第一，企业必须要完善管理制度，并且对各种制度有一个坚决的、到位的执行。我们研究后发现，有相当数量的“问题员工”是由于制度不够规范或者不够严谨而产生的。比如有些人总会把个人消费的票据，拿来作为公务活动报销，却能得到报销款。这显然是由于财务监管制度不健全或者执行不严格而造成的。因此企业必须要反思各项管理制度是否完备和完善，管理的各个环节是否需要进行梳理，是否存在疏漏。

第二，对“问题员工”要用其长，容其短。世界上没有十全十美的人，人的缺点往往可以通过一些适当的引导得以纠正变成优点。在管理时，要正确看待“问题员工”的表象，挖掘出他们的优势和强项，然后加以利用，当他们工作业绩表现好了以后，要给予他们恰当的奖励。对于他们的问题和缺点，首先给予一定的包容，在私下里与他们进行非常坦诚的、真诚的沟通，给他们一些有效的规劝和忠告；其次，督促和鞭策他们不断地进行自我反省、调整和改正。这其中，科学有效、奖惩分明的绩效管理体系会发挥极其重大的作用。总之，要给“问题员工”一些机会，不能一棒子打死，一概而论。

第三，站在管理者的角度总体来说，我认为没有不好的员工，只有管不好的员工。员工进入公司之后，如果出现问题却得不到好的管理和解决，那就应该是管理者的责任了。因此，管理者必须对“问题员工”引起相当的重视，沟通可以说是解决问题的一把利刃。

符俊平

UC优视公司人力资源部学习与发展经理

预判预防和对症下药

“问题员工”其实是相对而言的，其中更多的是指员工的工作习惯和价值观与企业文化和规章制度存在差异化甚至抵触性。不同企业的文化对员工的工作习惯、价值观都有不一样的期望和要求。例如，在工厂上班基本不容许员工上班迟到，但在有些互联网公司里，这根本就不是个问题，所以定位员工是不是“问题员工”需要结合企业文化和规章制度来看。通常，“问题”会集中表现在员工的工作态度、时间观念、沟通方式、待人接物等常见的工作习惯方面。当然，这些问题的产生有员工自身个性方面的原因，也有企业内部管理机制方面的原因，这都有可能催生“问题员工”。

对待“问题员工”需要把握以下几条关键的原则。

尽早发现尽早解决：“问题员工”表现出来的各种问题，对他自己，也是一种伤害，而且，也有可能直接影响到其他员工的工作情绪和积极性，所以要尽早发现、尽早帮助、尽早解决，避免“问题员工”病入膏肓，同时也能防止“传染”给其他员工。

避免逃避正视问题：有时候，由于担心“问题员工”滋事生非，管理人员通常也不

好意思纠正这些“问题员工”，对他们的行为往往是睁一只眼闭一只眼，只是希望他们不要把问题扩大化即可。这种逃避问题的做法不但不能解决问题，反而会让问题更加恶化。

对事不对人：与“问题员工”沟通过程中，要注意言语表达的方式，尽可能避免用“你总是……”、“你这个蠢货”等带有明显攻击性的言语。保持开放的心态，阐明你自己的期望和目标，以及“问题员工”对这些实现目标的重要性。另外，也要给予“问题员工”申辩和解释的机会。毕竟我们沟通的目的主要是为了使“问题员工”意识到自身的问题所在，这是他们改善工作习惯的第一步。

当机立断：当开放性沟通不能让员工的情况有所改善时，那么就需要采取“当机立断”的决策了。如果员工的“问题”已经超越自己的专业范围，例如员工心理治疗问题，就应该立即借助外部资源解决问题；如果员工的“问题”被确定为“顽症”时，那就应该当机立断地终止劳动合同。

具体来说，“问题员工”的管理措施可以分为“预判预防”和“对症下药”两个方面。

预判预防：我们在公司里面建立有效的信息沟通渠道，例如组织气氛调查、多种形式的沟通等，确保实时掌握员工心理动态；另外，结合公司现有的管理系统，例如员工考勤系统、项目管理系统等，可以从中获得“问题员工”的征兆信息，然后尽早采取相应的改善措施，避免情况进一步恶化。

对症下药：如果“问题员工”的状况确实存在，那就需要结合具体情况彻底根治了。在公正和公平的前提之下，通常可以采用的流程是：收集信息和证据→信息分析→制定沟通策略→员工沟通与交流→提供改善建议→观察和评估改善结果→反馈结果→终止合作。

幽　默

@第一使徒：《源代码》说的是一群程序员为了找出bug不断debug的故事。

@程序员的那些事：九点睡是村里人，十点睡是厂里人，十一点睡是校内人，十二点睡是官府人，一点睡是IT人，两点睡是IT人，三点睡是IT人，四点睡是IT人，五点睡是IT人，六点睡是IT人，总是睡不够的还是IT人。

@佚名网友：君要臣死，臣Facebook。

@StarKnight：朋友的Gtalk签名档一直挂着“少壮不努力，老大写程序”，今天终于弄明白，他的意思是手下的人都不干活，身为老大只好自己写程序。。。

@高焕堂：当你不知道情人的真实想法的时候，可别急着用debug方法去打开。。。

@玉伯也叫射雕：好的代码犹如炎热夏日里的一罐王老吉，可去浮热，可安心燥。一罐下口，气定神宁。

@windywater：世界上最尴尬的选择，是你是switch我是default，只有别人都不合适了，才能轮到我。

@裸奔的鱼：最伤感的莫过于while遇到break，if需要else。

@程序员的那些事：《程序员看感情》世界上最远的距离，不是生与死的距离，而是我在if里，你在else里，虽然经常一起出现，但却永不结伴执行。

@dllgwgy：路漫漫其修远兮，开着空调过夏季；找工作兮不容易，升职加薪大鸭梨。吾将上下而求索，边找bug边扎啤；安心coding不贰心，一头扎进代码里。

作者介绍：程序员幽默

新浪微博用户，博主为一名软件开发工程师，分享程序员相关笑话、趣图、观点、资料等。如果您有关于程序员的幽默故事，欢迎投稿，可私信或@程序员幽默。

微软的苹果香味

专访微软MacBU组成员

记者 / 木易

这是一只游走于微软和苹果两家公司、两大产品平台之间的特殊团队，与使用PC和Windows Phone的微软人不同，他们日常使用的是Mac电脑和iPhone手机，还时不时要向不理解“为什么要为苹果开发应用软件”的同事解释原因。在微软内部，他们被称作MacBU组。

今年6月1日，MacBU组年内最重要的一项产品——Office 2011 for Mac中文版正式发售，这是1985年以来微软推出的第一个基于Mac平台的中文版Office系统，新版Office for Mac首次加入了Outlook组件，引入了广受欢迎的Ribbon界面，同时力求兼顾Mac用户的使用习惯。中文版软件除具备英文版的全部功能外，还为中文用户进行了精心优化，如自带十多种中文字体库、支持包括竖排在内的多种中文排版方式与文字效果、拼音笔画排序和拼音汉字标示、支持GB18030汉字编码标准及五种少数民族语言文字等等。

尽管此前诸多的研发实绩——如为Windows 7、Office 2010、Exchange、Kinect等新品的成功开发做出了重大贡献，使Office 2011 for Mac中文版的面世就像是微软亚太研发集团漫漫长征途中迈出的小小一步，但它却比以往更清晰地展现出微软中国研发团队越来越强大的力量。

近日，《程序员》记者探访微软亚太研发集团，对话MacBU组核心成员，试图揭秘这支“特殊”的微软团队背后的研发故事。

历史：微软与苹果长期的竞合关系

作为数字世界两家巨头，苹果和微软眼下正在手机、PC、平板电脑甚至是云服务领域展开一场场激战。不过，或许大多数人只看到弥漫于一个个战区上空的硝烟，却不甚了解这两位巨人的合作史几乎与竞争史同样长久。

事实上，凭借图形界面（GUI）而大获成功的苹果Macintosh电脑和微软Windows有着一位共同的导师，那就是在施乐公司（XEROX）PARC实验室里诞生的Alto个人电脑样机（1979年，施乐公司允许史蒂夫·乔布斯率众参观PARC实验室。当时已配备图形界面的Alto电脑样机让乔布斯大开眼界。之后负责Alto研发的几位工程师很快被乔布斯挖到苹果，几年后以图形界面为卖点的Macintosh电脑问世。比尔·盖茨本人也对施乐公司PARC实验室推崇备至。后来在回应史蒂夫·乔布斯有关Windows“窃取”Macintosh图形界面设计的指控时，比尔·盖茨还曾轻松地揶揄说：“不，史蒂夫，我想那更像是我们都有一位名叫施乐的阔佬邻居，你破门而入想要偷人家的电视机，却发现我先你一步等在那里，而且你居然敢说：‘嗨，那是不公平的！我想偷那台电视机！’”）——Alto率先应用了鼠标、图形界面及以太网络接口，并可以和另一台电脑及打印机互联工作。在某种意义上，苹果和微软可说都是PARC实验室创新精神的传承者和发扬光大者。

少为人知的是，微软办公软件Office的Mac版本要比Windows版本诞生得更早一些。早在1982年初，IBM刚刚推出个人电脑还没多久，微软已和苹果签订了为Macintosh电脑创作办公应用程序的协议。1985年，Word和Excel登陆苹果Macintosh电脑，并获得了史蒂夫·乔布斯及苹果电脑用户的一致认可之后，Office for Mac从未停止过更新和升级，并一直被众多用户认为

是苹果电脑上最具实用价值的应用软件之一。

而微软MacBU组聚集了一群身心都在微软却专注于苹果相关产品开发的人，这支团队的历史悠久，可上溯至上个世纪八十年代的DOS时期。他们的工作正是微软竞合关系的一线战场。

竞争焦点：IT消费者化

近年来，“IT消费者化”潮流大行其道，苹果公司在手机、平板等领域表现惊艳，而微软似乎步步落后，错过潮流。对此，微软全球资深副总裁、微软亚太研发集团主席张亚勤认为，“一是IT技术快速渗透到消费领域，从而改变了消费电子市场的竞争格局。像杀入手机市场的苹果、谷歌、HTC，原本都具有深厚的IT技术背景。二是面向消费者的推广变成一种主流。尽管这一两年来在面向消费者的终端产品上，微软丢掉了一些机会，但在‘IT消费者化’的下一轮竞争中，微软会加快速度。我一直认为，IT业的‘三大战役’才刚刚开始。”

张亚勤所说的三大战役是指未来产业竞争的三个焦点：一是云计算，微软已初步确立起竞争优势；二是PC和移动终端的拉锯战，截至目前包括苹果在内还没有企业能对微软在PC软件领域的领导权发起挑战，而Windows Phone 7的推出则已然令全球移动终端操作系统战场风云突变；三是架设于“云”和“端”之上的社会网络平台——当前Xbox Live已“织造”出了全球最大的游戏社交网络，从去年开始备受消费者追捧的Kinect很可能会让微软在此领域的领先地位更加巩固。

MacBU组的产品经理（Program Manager）张毅从另一个角度解读了“IT消费者化”，之前在微软Exchange部门做产品开发，张毅由于“很想做与终端用户相关的产品”，并且和MacBU组的几位创始人很熟稔，所以在一年前加盟了这个充满活力的团队。他的看法是，在苹果复兴之前的那十几年里，许多IT公司特别热衷于为企业级市场开发产品、提供服务，“这是因为企业客户的忠诚度相对较高，当他们适应了某种平台、软件和服务，便不会轻易地汰旧涤新，毕竟新事物有很大的不确定性，也需要员工花费时间成本去适应。而苹果的创新让大家了解到，企业客户使用的电子产品或IT技术都是可以和消费者的需求及偏好相结合的，企业的员工也是消费者，工作的时候你要扮演职务角色，生活的时候你还是你自己。你可以选择你喜欢的产品和服务。所以不是说针对消费者开发的成功产品就不能被带入企业中。微软已经意识到这个趋势，所以目前我们也在努力调整航向，希望在不久的将来把惊喜带给消费者。”他说。

跨界与融合：向苹果学习什么

现在，你可以在Office 2011 for Mac中文版的使用中体验另一个微软产品面孔。此前国内Mac电脑用户虽然也可以选择iWork、OpenOffice等办公软件来处理日常工作，但毕竟在同事、客户中使用PC和微软Office的人占大多数——文档兼容性成问题，在Mac电脑上辛辛苦苦排好的格式往往存到PC上就完全走样；当然用户也可以使用Office for Mac，然而过去很长时间里，这款重磅软件都没有官方中文版。现在，Office for Mac 中文版已经上市。

微软Word项目经理组长、同时负责整个MacBU组用户体验的邵汉仪已在微软工作了11年，他一直为“结合微软和苹果两家公司的精华”、让产品既具备微软“智造”的强大功能又拥有苹果的典雅设计和人性化体验而求索。“苹果的DNA是设计，”他说：“最能代表苹果产品与众不同之处的是设计。我们要想将基于苹果平台的产品做到尽善尽美，就一定要达到、甚至在一些地方要超过苹果，所以要比苹果的工程师更加重视界面和功能的平衡。”

或许不少新员工不能透彻地理解MacBU组对微软的价值，但今天的微软的确比以往任何时候都更需要这支懂微软、也懂苹果的团队——MacBU组的长久存在和持续壮大既体现出微软对于行业其他杰出企业的尊重，也有利于微软揣摩消费者的需求演进趋向、并用以指引未来的产品革新。史蒂夫·乔布斯曾经援引加拿大传奇冰球手韦恩·格雷茨基（Wayne Gretzky）的一句名言来解说苹果公司的经营和创新方针：

“我不会去追寻冰球飞过的轨迹，而是径直滑向冰球的落点”。这句话现在被邵汉仪用来阐释MacBU组的目标——“微软在办公软件领域也已是寻找冰球落点的领先者，但如何在产品使用的过程中、在用户体验方面也能做到持续领先，这是我们所强调的。这次Office 2011 for Mac的研发进程中，我们确实点燃了很多新的灵感，当我们发现一些想法前所未有时，就会全力以赴去实现。我觉得，苹果对用户体验的定义和解释是成功的，但这方面‘成功’不一定非得由苹果来界定。微软也可以是界定者。”

中国团队：善于独立解决各种难题

MacBU中国团队首席开发经理李生勇是这支团队的创建者，2006年，在美国业已功成名就的李生勇做出了一个不算艰难但却值得尊敬的决定。他对微软公司Mac事业部总经理艾立克·威尔弗里德（Eric Wilfrid）说，是时候在中国建立一支MacBU团队了，因为在中国有着许许多多优秀的人才，必须去尝试把握这种智慧资源集中的优势。艾立克认可了李生勇的建议，于是李生勇单枪匹马回到北京，带着四名大学生，打造了微软MacBU组的中国分支。

当时李生勇来到北京带回来的仅仅是一个“远程桌面连接”的小项目，而在5年后Office 2011 for Mac的开发进程中，不仅团队规模扩展了十倍，其所负责的项目的重要程度也较以往有了天渊之别——微软总部决定将整个Excel 2011交由中国团队完成，原因是分布于世界各地的每一支团队都有着鲜明的特点，而中国团队的特点是“善于独立解决各种难题”。正因如此，新的Excel组件从功能设计到创新灵感的实现再到产品的测试都是由北京的这支小团队来完成。值得一提的是，由北京团队倾力打造的一项Excel 2011新功能赢得了先期试用者的交口赞誉——支持自动生成的数据透视表能够有效地简化新用户的操作步骤，并极大地提升了Excel电子表格的专业感及表现力。

李生勇认为，中国浩瀚的市场规模是任何企业都无法忽视的。根据摩根士丹利和AlphaWise于今年3月发布的一项调查结果，如果苹果的iPhone在价格上占有优势，那么它很可能会获得53%的中国3G用户市场份额。另外，市场研究机构IDC的统计数据亦显示，如今中国已成为全球最大的手机市场、汽车市场，并将于2012年超越美国成为世界第一大PC市场。很显然，当最看重用户体验的MacBU组遇到最具规模和潜力的中国市场，可以期待两者的化学反应对微软亚太研发集团未来创新战略的影响。

新目标：中国成为微软全新用户体验的创造中心

微软亚太研发集团MacBU组的一系列实践表明，以中国为基地的技术创新和跨区域研发合作完全能够取得辐射全球用户的效果。五年前，和研发集团的许多同事一样，李生勇及MacBU组的所有员工都不仅仅是因为中国的市场规模大而选择留在中国搞研发——“因为我们喜欢中国，所以才扎根中国。我的同事中很多外国人，但他们和我一样，首先是喜欢中国，其次是喜欢我们开发的产品。”

曾就职于苹果日本公司的理查德·斯普瑞格（Richard Sprague）目前任MacBU中国团队的项目总监。他在1997年就已加盟微软，五年前他来到中国，为的是“找回在Mac平台上工作的感觉”。他认为由于中国市场的绝对容量足够大，基于苹果Mac电脑的软件可能会像十多年前的日本那样，迎来快速的增长：“我刚去日本时，苹果的市场份额很小，但增长很快，只是那时基于苹果电脑的应用软件不但少、质量也不高。现在苹果产品在中国也逐渐打开了市场，中文版的Office可说是生逢其时，它不仅能满足Mac用户的办公需求，还能切实帮助他们提高工作效率——这种感觉非常酷。”

2011年5月25日，在微软亚太研发集团总部新大楼的启用仪式上，张亚勤表示：“中国对微软而言，不止是研发中心，还是战略中心。”不仅如此，中国还很有可能成为微软创造全新用户体验的中心，而MacBU中国团队则是这个未来的核心推动者。

手机游戏传播的整合尝试

文 / 郑金条

破茧而出的喜悦和忧愁

开发者可能会对作品的期待并不尽一致，但当一款作品倾尽开发者心血并最终成型即将推出市场的那刻，总会有相近的阶段性的成就感和喜悦。

“瞧，这小破孩把咱们折腾的，可是您知道，现在我简直开心得想哭了。”

一款能够对自己交代的作品，足以暂时宽慰以前日日夜夜昏天暗地的付出。

可这并不是最关键的部分，此刻你只是把一款未经市场认证、还未从用户那里获得回馈的待验证作品带到了一个静默无闻的角落。

关键在于你要有足够的布局和机会，或者找到一个能够协助你完成应用行销的专业发行公司和你一起，把这款你认为可以经得住用户推敲的游戏推广到用户面前，然后虔诚地接受玩家的褒扬或者呵斥。

在前的起点和在后的起点

对部分独立开发者而言，往往需要默认一个事实：在低成本营销的名义下以口碑的方式和用户做每一次近距离的接触。

在应用市场的现阶段，强者愈强（资源和资本型优势者）的现实是每一个人都需要面对的残酷问题，很自然地事态就会演变为需要在数十万的应用堆里寻找一寸生机，经历艰涩的自我挣扎的冒险历程。

那些能够像Cut The Rope从一开始就依托Chillingo超强的发行优势一跃成为全球用户下载榜单的热门；像Tap Zoo可以付费依托Tapjoy的营销水准迅速累积用户适用基础；像Gameloft与韩国LG公司或者Zynga与AT&T一样布局预装市场拓展激活数据，属于“在前的起点”（有资源和门道）。

而作为白手起家，完全缺乏营销资源和经验的开发者则完全属于后者，是相当彻底的“在后的起点”。可能每个人心中都有一个大体的营销图谱，但因为资源受限，游戏从一开始阶段就将遭遇到面向未来不知所措的挑战。

“我们有一颗无比坚定的心，可是我们同样很茫然。”

我和你，才是最佳渠道

低成本营销并不是一条末路。

我们可以试图去还原下一般产品从制造开始，然后经过了广告包装之后，最终的目标走向还是面向用户，就意味着说产品产出最后的价值在于消弭和终端用户之间的信息差，并达到用户手上实现消费。

在这层表述上我们能够看到，所有中间环节的努力，包括4A公司的理念包装或者媒介机构的展示推动都只是在于消解隔阂、到达用户和实现转化，而这个和我们之前的分析“在前的起点”和“在后的起点”基本上相似，资源型的行销可以借力上位，而低成本的行销则只需要找到一座避开信息差的桥梁就可以实现和用户之间的无缝对接。

“从黑暗到黑暗之间，隔着一层光亮。”

在社会化媒介主宰的时期，个体用户同样是最佳的媒介渠道。

每一个人都是一个自主的信息源，并且是一个不安分的试图时刻和外界产生关联的信息源。此刻所有的信息接触已经从被动（并且被

意念灌输）模式直接调整为用户自我创造和遴选自我中意的信息。

这种进程可以展示为：

论坛（以天涯社区为例），以信息为主体，面向所有阅读者的公众化信息，信息展示以用户评论推动重复曝光频度。

博客（以博客大巴为例），以作者和信息为双重主体，用户评论不再具有重复推动曝光的功能，信息转向半私有化，幅度从面向论坛版块选择性压缩为只面向博客作者。

社交网络（以Twitter和新浪微博为例），以分享者为表象关注方式，以具体信息为主体，以转发形式推动信息的无限制扩散，信息为半私有性质，只面向关注圈展示。

可以看到在用户自媒体的范畴下，低成本行销同样以直接面向目标用户为目的。并且从上述的解析中可以看到，自媒体渠道其实完全拥有了病毒化的属性，这种属性和我们上文所提示到的“在前的起点”所驾驭的资源和行销步骤与方式完全不一致。

对于饱受行销渠道和资源限制的开发者而言，完全可能是一扇关了门之后打开的窗。

用户就在身边

和传统行销依托广告公司布局媒介渠道的方式相比，社会化营销几乎已经消弭了让广告主潜藏在包装表演背后的那种隔阂感。不管是在Facebook、Google Plus、Twitter抑或者在新浪微博上，开发者本身就是一个鲜活的营销载体，身上的每一个元素都充斥着游戏出彩的环节甚至包含着开发者那种独特的深情。

“扯开了面具，卸下了浓妆，彼此以双手触碰感觉。”

此刻，信息所到之处已经不仅仅只是一群躲在屏幕后面只想以概念来识记品牌的旁观者，而是转化为带着情感共鸣的真实消费者。

那种用创意让无数人一起奔走相告的喜悦不仅仅只是属于神秘莫测的4A创意群体，社会化网络真切地让创意脱离臆想（从揣测用户需求出发，事实上更多是广告主或者广告公司角度），融入更多交互当时的情感和即时创意火花。

在类似新浪微博这样的传播平台，好的创意价值信息并不归属于任何人，而是从属于所有期待一起分享观点见解的用户群体。如我们前面所分析到的，在微博平台上信息发布者并非关键因素，所有的引爆点都只在于信息内容本身，不需要发布者有多庞大的关注群体，而只是需要一个合适的用户转发来点燃一条价值信息的生命力。

这里任何一个试图主宰营销都是无力的，社会化微博里所有的凝聚点都在创建有价值的信息。

“生硬和无聊的广告信息腐烂在杜撰者手里，有共鸣的创意将走遍微博的各个角落。”

在社交化平台（微博）上，每一个人/账号都是弥足珍贵的信息传递环节，同时也是超强的过滤终端，不值得分享的内容在任何一个角度都可能夭折在用户的漠视中。每一个用户/账号都具有信息甄别和对情感喜好的传达功能，违背交互初衷的简单广告就很难获得来自用户方的认同和支持，而乏味的内容则同样致命。

创意布局和星火燎原

任何的规则都是相似的，开发者在自媒体的环境中，首要的也是针对定向用户做创意价值的分享。

“好的创解任何时候，都是多米诺骨牌的起点。”

在没有人了解或者仅有少部分人（甚至这部分人并没有出现在分享的平台上）了解游戏的情况下，如何在静寂无声的网络世界里做一声低沉的呐喊？

假设你自以为满意的那款游戏以捉迷藏为核心素材，你如何以唤醒用户对这些既有印象的追忆，并最终将用户的印象情感导向开发者的游戏？

在新浪微博中，以昭告天下的方式得意洋洋地推介了一款捉迷藏的游戏，很可能我们只是在无数的信息垃圾中再可耻地增加了其中的一条而已。就像我们前面提到的，即便你有足够的关注者，但也应该相信大多数只是对广告信条熟视无睹的过滤用户，真正活络的营销在于发动用户投入自身的情感将他自己以及他周围能够接触到的

用户紧紧地捆绑在整个营销周边。

一个可能性的假设（以新浪微博为例）：触动用户对捉迷藏事件的回忆（足够的群体和事件共鸣性），然后将对事件的回忆转向对体验相似模式游戏的诉求（不一定直接促成下载，但是这种印象激发随时都有可能实现用户的转化）。

事实上在社会化营销优化（SMO）的界定中，具有吸引力的呈现至关重要，这不仅仅是一种策略考量，而是实在地触碰到用户的心灵感受。其中又可以区分为两个层面，创造吻合用户内在需求的价值内容以及驱动用户创造他们足以影响他自己圈子的价值内容。并且，由用户充当内容创造主体和传播主体的传递模式才是实现营销价值最大化的关键所在，也是社会化营销利用社交媒体本身交互实现的价值最大化。

我们假定的这款捉迷藏游戏，如果能以“谁的方式最有趣和印象感触最深”作为整体的切入点，将有极大的可能撬开用户积极参与的意愿。而开发者作为信息发起者更可能充当的只是一个信息主持的身份，并在适当的时候退居幕后，欣赏整体信息传播链条上每一个多米诺骨牌节点的故事。

博弈：社会化价值呈现的动力

个人自媒体能否真正实现价值拓展可能一直充满疑惑，没有专业4A的创意架构，没有超级覆盖率的媒介支撑，没有足够雄厚的资源和资本做运营的财力自持，社会化是单一个人就能够实现信息传播的最大化吗？

“我只是孤身苍茫一人，他们甚至不愿意和我擦肩而过。”

以新浪微博为例，有一个需要澄清的事情：单个帐号的关注者并不属于帐号本身，而是从属于平台和他们到处窥探奔走的信息。

拥有关注者的多寡只是在于用户认为该帐号和他们目标信息的关联度（明星粉丝和无聊八卦除外，这点和营销以及商务属性关联偏小）。一个有价值的创意和信息，可以不需要基础关注者，而只需要一个挖掘点就能够支撑起它本身对整体网络各个节点的冲击。

这就有一个很明显的幅度区别，传统的渠道封闭而不得不依托广告行销部门和他们对媒介的驾驭能力间接实行的传播在自媒体映像中已经开始显得不合乎规则。

“老伴，又广告啦，赶紧换台。”

因为作为开发者，你对自己游戏产品的特征和用户需求假想将比其他的环节更加了如指掌，不需要凭空捏造需求，也不需要在各种渠道之间角力去搜寻用户，用户本身已经真切地布局在交互媒体上，并且更加真实（各种关系链和情感投入）和主动。

在没有巨额营销经费投入的节点上，可以相信用户一方面痴迷着品牌的概念，另一方面又开始觉得真实接触比空泛的概念更具意义。

社会化为个人的低成本行销打造了前所未有的机会，就如我们前面所解析的任何一个信息源都有可能成为价值传播的多米诺骨牌起点，而所有的关键只是汇集到如何产生这个让全体一起情感狂欢的价值源。

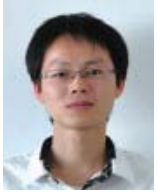
营销正在发生变革，媒体时代将被交互平台无限解构。

仍然以我们前文提到的捉迷藏游戏为例，时代和环境正在发生变化，过去曾发生过的这些趣事在目前的现实生活中已经很难重现，包括场景完全消失了，现在人的娱乐需求也改观了，就意味着这种儿时的回忆将无比浓烈，甜蜜甚至惆怅。人的情感和需求本能在不同族群的思考中是相通的，从办公室里诞生的创意不一定能够及得上交互当刻的情感流露。

微不足道的总结

对于只能依赖低成本进行营销的开发者而言，社会化媒体是一个强势助推力。

“在偏远的乡下，没有自来水的村民自发掘出的井水更加清甜。”



郑金条

游戏邦负责人，游戏邦主要关注和解析国内外社交游戏和手机游戏领域，并定期做深度行业阐述。

责任编辑：常政（changzheng@csdn.net）

“不替客户选择”

三星（中国）投资有限公司副总裁周晓阳专访

记者 / 杨鹏飞

仿佛一夜小楼春雨后，智能手机圈内便发生了剧变。三星如何迅速崛起？bada在三星的战略地位中有多重要？智能电视和平板电脑是什么关系？带着这些疑问，记者采访了三星电子副总裁、在线业务部负责人周晓阳。

周晓阳：三星电子副总裁，中国三星在线业务负责人。在中国无线通信领域已有近20年的经历。
图中右手边物品分别为Galaxy平板，三星智能手机和智能电视遥控器

记者：三星的产品里您最满意的一款设备是什么？

周晓阳：到目前为止，我觉得还没有完全让我满意的设备，毕竟手机和平板无法合为一体。虽然也有人用三星Galaxy平板打电话，但终归不如手机的体验来得舒服；而手机由于屏幕的原因在处理一些事务的时候又无法达到平板电脑那样的效果和体验。也许有一天，当手机可以任意改变大小时才能真正令我满意吧。

记者：作为目前世界上第二大智能手机厂商，三星的优势是什么？

周晓阳：三星的优势就是“集中”和“专注”，一旦认准发展方向就迅速做出行动。三星并不是最早做智能手机的厂家，但是一旦确定路线，便以最快的速度行动。当然，这也跟三星是一家综合电子制造厂商有关。我们有零件、显示、存储、通讯系统……这一整套产品，确保了三星的竞争优势。早在20世纪90年代，三星就开始做手机。在2000年的时候，已经占据了韩国50%的市场份额，然后开始发展全球市场。

记者：开发者更关心平台，那么Android和bada，三星从战略上更侧重于那一个？Windows Phone 7有计划做吗？

周晓阳：我们的原则是“不替客户选择”，不论是面对开发者、运营商，还是消费者，我们提供多平台解决方案，让客户自由选择。我个人认为，短期来看不可能出现单一平台“一统天下”的情况，一定是有各种平台之间不断地竞争。因为每个平台的出发点不同，当一个平台暂时变成第一之后，总会有新的平台起来挑战它的地位。如果用户清楚地知道自己想要的是什么，我们会尊重客户的意见。但是如果客户还不清楚自己想要什么，我们会向他推荐三星自己的解决方案。至于WP7，只要客户需要，我们就会去做。三星这么大的一家公司，有实力同时支持这么多的平台，当然包括bada。

记者：三星的应用商店目前情况怎么样？

周晓阳：三星是一个综合电子制造厂商，整个核心就是硬件的设计和制造：怎么设计出最好的电子产品，然后进行最高效率地制造，最后以最适合消费者的渠道将产品分发出去。但近两年来，通信已经不仅仅是语音通信，一款手机仅靠外形和通话功能已远不能满足消费者的需求，于是三星对应用开始重视。因为所


新款三星Android平板摄像头非常清晰

有的设备包括手机、平板、智能电视都需要，所以三星在去年1月份成立了在线商店。目的也非常简单：旨在为用户提供更好的解决方案。

记者：bada在全球的销量如何？应用如何付费？bada今年会有什么动作吗？

周晓阳：3月份的数据是将近一千万的出货量。bada在线商店的付费系统已经有了，广告系统正在搭建，当然我们还是希望能够由第三方来完成，和更多的广告商进行合作。2011年下半年我们会有4-5款新机型发布，而且也会进行新一轮的应用开发比赛，来吸引更多的开发者。和以往不同的是，本次大赛将会加入对跨平台支持的评比。目前已经看到有开发者挣钱了，成都一名大学生做的一款游戏到现在下载次数已经超过1万多次。

记者：bada开发平台有什么新变化吗？

周晓阳：bada平台现在支持C/C++，随着SDK的升级，开发的步骤会越来越简单。最近，我们在几所大学做了个项目，发现学电子的大一学生就可以做，他们基本都没学过高级语言。所以说，只要工具做得好，是不是精通C我觉得已经不是很重要。目前bada平台已经很成熟了，接下来的任务就是构建自己的生态系统。对于开发者来说，一定要在开发初期就考虑好做跨平台，软件结构要尽量简单，也要注意屏幕适配。

记者：能否介绍一下Smart TV？

周晓阳：电视经历了一些发展过程，如追求超薄、高清、平面、3D等，但到了智能时代，这些还不够。像手机、平板，基本上是个人独享，你很难和别人聚在一起分享。但是电视不同，一起看电视的人组成的圈子，成员之间的关系要远比社交网站形成的圈子里成员之间的关系更加密切。至少在中国是这样。

智能电视，不仅仅可以看电视节目，还可以去阅读、分享、玩游戏。通过无线模块连接平板、手机，可以随时随地看电视和分享信息，还可以将平板或者手机上的游戏通过智能电视的高清大屏幕显示，这样玩起来更爽。

智能电视采用5mm边框设计，可以使画面更大，视角更宽，享受高品质的2D&3D画质，厚度仅为29.7mm。通过WIFI连接网络，通过自带浏览器浏览网页，通过三星官方的在线应用商店下载和安装应用，包括开心、豆瓣、大众点评、新浪微博等各种应用，将社交化网络带入客厅。智能电视基于Linux内核，Linux开发者可以很方便地进行基于Linux操作系统的三星智能电视应用开发。同时在这里告诉各位开发者一个大好消息：三星正在启动的2011年应用开发者挑战赛——“星空大赛”，奖金总数高达200万元！参赛选手有望拿到巨额奖金和bada最新款手机，详情请关注大赛专区。

大赛网址：http://dasai.samsung.com.cn/

基于Linux操作系统的三星智能电视

相关资料

bada：bada在韩语里是“海洋”的意思，是由三星公司自行开发的智能手机平台，于2009年11月10日发布。底层为Linux内核，为许多拥有丰富功能和良好用户体验的软件应用提供支持。bada重视SNS集成和LBS应用，以灵活的配置和极佳的用户交互性为特点。

三星Smart TV：三星2011年推出了Smart TV这个新概念，将之视作家庭娱乐中心。三星Smart TV提供与智能手机和平板电脑互联互通功能，并提供Smart TV专用的在线应用商店，可以满足用户下载和安装应用程序的需求，同时满足电视大屏幕高分辨率的要求。底层基于Linux内核，Linux开发人员可以很方便的进行智能电视应用的开发。

瞄缺口，聚人力，深挖井

论网易iTownSDK开发平台

自苹果应用商店向第三方开放大获成功开始，开发平台就成为IT业界的热点，国内众互联网公司紧跟世界潮流，在近两年也陆续推出开发平台。而在众多国内开发平台中，网易iTownSDK显得有点特立独行：它是一个客户端网络休闲游戏的开发平台。是什么原因让网易选择建立这样一个开发平台？这个开发平台又有怎样的特点和前景？我们将对此一一分析。

休闲网游需求缺口大，网易布局平台深挖井

“宁挖一口井，不挖十个坑”是网易CEO丁磊的格言。在网易自行研发的大话西游2取得辉煌成绩后，多年以来，网易一直致力于深耕网络游戏市场，在MMORPG各个细分市场都推出了自己潜心研发多年的拳头产品。在MMORPG市场成绩耀眼的同时，在网络休闲游戏市场上，由于以往受制于市场规模，网易一直没有大动作。而近两年来，随着《植物大战僵尸》等休闲游戏的火爆，加上iOS、Android等移动智能平台设备的广泛推广，休闲游戏成为了众多游戏玩家的娱乐首选；更有不少以前从不玩游戏的人也开始玩起了休闲游戏。休闲游戏需求在迅速扩大，而占其中一大部分的网络休闲游戏，市面上的游戏数量以及玩法却远远没有跟上玩家的需求。《大众软件》在2010年中国电脑游戏产业报告中指出，在2010年运营的网络游戏中，休闲类网络游戏仅有32款。就是因为看中了这一需求缺口，网易建立了iTown这一网络休闲游戏平台，并通过开放功能强大的SDK，让第三方加入开发，共同深耕这一市场。

开放自主研发引擎，提供优质配套开发方案

我们曾就国内游戏开发的现状对游戏开发人员以及团队进行了调研，发现曾经尝试进行客户端网游研发的团队，尤其是中小开发团队，其最大的难题在于无法找到良好的游戏开发整体解决方案。碍于资金等问题，这些团队基本上都是选择免费的开源引擎，免费的界面、音效解决方案等搭配进行开发。而由于这些开源解决方案缺乏专人维护和针对需求继续开发，很多设计往往无法找到匹配的功能实现。所以即使能够开发出最终成品，其游戏质量、稳定性都是不能达到应有要求的。

作为客户端网络游戏开发平台，网易提供的一整套游戏开发解决方案包括：自主研发5年，并成功运用到《梦幻西游》等网易大型成熟游戏上的NeoX引擎；采用了《星际争霸2》等众多大型游戏所使用的收费解决方案GFX，还有被誉为全球最好的音效系统FMOD，以及数据存储、网络、服务器架构等等经过多年验证的方案，配套对应的开发工具，第三方均可免费使用。而网易预先设置的好友、聊天、俱乐部、赛事等游戏周边功能则节省了第三方的大量开发工作，使其开发成本大大降低。

对第三方极力扶持体现平台重要地位

自iTownSDK开发平台推出以来，网易就一直强调该平台是网易在休闲网游市场的“战

略布局”、“重点产品”。在7月3日举办的iTownSDK百万奖金游戏开发大赛启动仪式上发言以及接受采访时，网易游戏副总裁丁迎峰也重新强调，iTown游戏平台是网易在网络休闲游戏市场上长期性的唯一重点产品。当然，单凭一公司高层的现场表态，我们不能确认iTown这一游戏平台对网易是否真的如此重要，在结合了iTownSDK开发平台对第三方的扶持政策，以及近期所举办的两个开发比赛的投入后，我们才比较有理由相信这一事实。

iTownSDK的奖励分成制度，从平台介绍中看到，在第三方的游戏并未盈利的情况下，iTownSDK会对达到上线标准的游戏首先付1万元以上的上线奖金，以对第三方的投入作出一定的回报保障；游戏上线后的利润分成也达到前所未有的2:8（第三方拿大头）；更会根据第三方游戏的每月在线人数情况给与奖励。

可以说，在运营第三方游戏上，网易几乎是不赚钱的。“真有这样的好事？网易不赚钱凭什么做这样的平台？”活动采访中我们提出了这样的质疑。而网易游戏副总裁丁迎峰的回答是“靠社区赚钱”。原来iTown除了是个聚集各种休闲游戏的游戏平台外，还有一个3D社区世界。iTown通过第三方开发的各种新颖游戏以及3D社区，和其他休闲游戏平台作出明确的区隔。网易通过3D社区对iTown的玩家进行沉淀，并将通过在社区提供更多增值服务来盈利。

网易iTownSDK为第三方开发者提供了丰厚的奖金

iTownSDK对第三方团队开放资金资助申请，被认为是又一大力扶持政策。其实国内推出的开发平台，大多附有专项的投资基金对第三方进行投资。然而出于风险考虑，其投资审核条件都设置得比较高，一般只针对实力强大的团队、公司进行投资。而iTownSDK对第三方团队的资金资助将不设高门槛，无论团队或公司大小，只要有达到质量要求的游戏，都可以进行申请。网易会根据游戏质量决定是否给与资金帮助。在这个政策下，中小开发团队将得到与大团队一样的扶持和机会，尤其是对初创团队而言是一个相当有利的创业环境。

最后一个iTown平台作为重点战略产品的佐证，则是iTownSDK近期举办的两个开发比赛。4月16日，iTownSDK宣布举办开发作品征集活动，前5名的作品无论是否达到上线标准，均可共享12万元的活动奖金。在活动结束不久，网易立即举办第二波总奖金100万元以上的比赛，一个游戏最高可获得30万元以上的奖金。在iTown平台尚未对外运营产生任何盈利的情况下，网易就一次比一次投入更多的奖金对开发者进行奖励，其投入决心可见一斑。通过上述的政策和活动，我们可以看到网易对iTownSDK的第三方是作出了很大的扶持力度的，其“战略重点产品”的地位应该可信。

经过以上分析，我们对网易iTownSDK这一开发平台的归纳就是“瞄缺口，聚人力，深挖井”：在瞄准了客户端网络休闲游戏这一需求缺口后，通过优质的游戏开发解决方案以及大力的第三方扶持政策，吸引第三方团队共同合作建设游戏平台占领市场，成为细分市场的龙头。只要第三方足够给力，以网易一惯的精品战略，配合多年的网游推广经验和能力，平台的前景应该是相当广阔的。

编者按：湖南工业大学青年教师田飞设计的Line Phone概念手机是2011年芙蓉杯国际工业设计大赛“数字产品与服务设计类”金奖获奖作品。Line Phone概念手机完美结合了中国风和科技元素，部分资料（主要是指视频）在优酷上公布后，令人大为惊艳，并引起了设计、消费电子、互联网界的广泛关注，本文中，田飞将和大家分享关于Line Phone的设计心得。

Line Phone概念手机的设计感悟

文 / 田飞

概念手机：Line Phone

Line Phone是两年前就有的想法，一直断断续续在做，时值芙蓉杯正好启动，时间契合，所以就参赛了，运气很好，拿到了金奖。我想从以下几个方面来介绍我们的心得。

更简单的使用

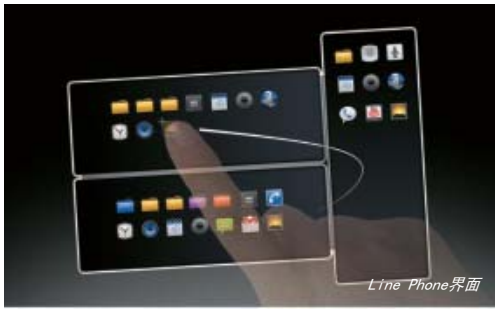
智能机的发展使手机的功能越来越强大，与此同时，其复杂性也不断增加，很多功能必须经过一系列深度的菜单操作后才能找到，所以在Line Phone最初的概念中，我们所设想的是，如何能将最常用的功能直接调出？而且这种常用功能的操作应该是符合人的认知习惯，使用户几乎不用学习，就可以记住这种操作。

于是我们尝试将设备的四条边线设计为触摸感应器，在边线45度斜角面滑动操作，可以进行相关情境下的相关动作。如下边线滑动代表缩放语意，那么在图片环境即缩放，音乐即音量，想想看，照相机情境下，下边线滑动自然就是镜头推拉，所以，在熟悉了具体的功能语意后，应用操作当然会更简单。

简单的另一部分是分享方法更简单。在设备靠近时，四边的感应器相互识别，将询问使用者是否配对，如选择是，设备周边的亮光会将设备包拢，以使其进入分享模式，在这种模式下，所有的信息，资料都可以随意拖拽进行交换。

沟通的目的是线下聚合

网上很多留言对“多设备组合看电影、玩游戏”这部分功能感到奇怪，认为谁会带这么多手机一起来用呢？事实上，这是将几个人的设备拼在一起来使用。我们鼓励人们聚在一起时，能“一起玩设备”，而不是“各自玩设备”。这个想法的出发点是因为我们发现：如今的网络越来越发达，而手机与人的距离越来越亲密，我们似乎习惯了通过设备和网络来与朋友进行沟通和接触，甚至在我们面对面一起聚会、吃饭的时候，大家也时不时刷一下开心网、微博，在SNS上给朋友一个微笑和触摸，却忽略了近在咫尺的真实的人。这是移动互联网带来便利性的同时，无法避免的问题所在。


Line Phone界面

可以想象在未来，在新一代伴随着网络成长的用户身上，这些问题将更明显。因此，我们的设计应该要考虑到这些社会问题的存在，应该鼓励和提醒用户，我们使用网络所做的所有沟通，最终的目的应当是线下的、真实的聚合。

文化对产品的影响

我们都明白，随着现代社会的发展、技术的进步，传统的文化、方法、记忆都无可避免地将会被有意无意的遗忘，这将会导致人与文化母体之间的一些联系被切断。所以，当代人为了体验、传承我们的本土文化，正在做出很多努力，比如翻修各种古建、身着汉服、强迫孩子们背诵《弟子规》、举办各自盛大的祭祖大典……

Line Phone模拟方向盘

Line Phone某种程度上就是试图用设计的手段来对此作出某种补偿，我想这种补偿的方法应该更多关注的是地域文化属性的根源和骨性，而不是其表象。所以，我们想要通过人们常用的设备操作，来唤醒哪怕一点文化体验的记忆，使其能和现代生活方式进行潜意识中的桥接，让文化的脉络在社会发展过程中自然地传承（非强迫、无意识地传承一部分文化的特性）。对我们来说，这就已经值得满足了。

Line Phone的设计灵感来自一位书法家朋友胡紫桂的作品，每次看到他草书作品中所包含的令人激动的传统与现代混杂的特征，都使我意识到这些美好的文化正在离我们越来越远。所以我们的设计就试图用书法中线条的概念来完成硬件感应器、界面，用模拟笔划的方法来设计操作行为等。

以上就是关于Line Phone的一些想法。此外，还想聊的是关于创新。很多人把创新和设计联系在一起，认为设计可以给予更多的创新

Line Phone的设计灵感来自书法作品

支持，我想这是对设计师的赞誉。其实，创新无处不在，特别是在今天的移动互联网热潮的大环境下，更是如此。我能够分享的是关于我们作为设计师的创新方法。

设计是一门交叉学科，设计师需要涉猎的知识面非常广，并且要有良好的理解力，能真正理解用户，他同时扮演的是一个超级多面体般的用户。

但产品设计不是科幻，所以我们同时需要对技术和可实现性要有深刻的理解，对材料、工艺、新科技都要有所理解，只有这样才能和工程人员进行有效的沟通。在信息化时代的今天，很多产品是以软件、应用的形式出现，就更要求设计师要理解程序、理解数据、理解机器和设备的读写流程、理解软件工程的思想。

但设计师在理解技术的同时，应该将技术视为一个“黑盒子”。我们只需要知道输入和输出分别是什么，然后为技术去寻找更合理的商业应用。这种寻找商业应用的过程，其实依靠的是对用户真正需求的把握。今天的时代，大量的信息、科技、机遇交叠穿插，需要我们去理解和把握它们，并为用户创造真正的价值。

田飞

设计学硕士。现为湖南工业大学讲师、湖南省产品包装工业设计中心研究员。设计作品包括“Line Phone”概念手机、通用电气Edison奖亚太区07颁奖礼影像及流程设计等众多项目。

责任编辑：常政（changzheng@csdn.net）

感知世界 触景生情

增强现实技术简介

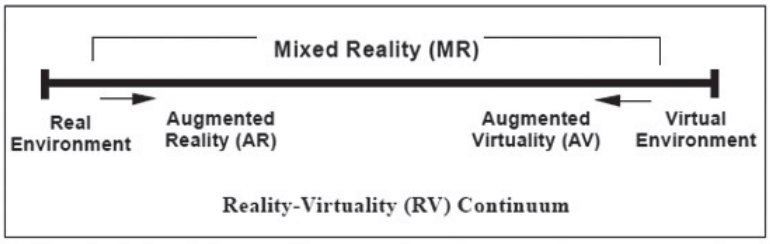
文 / 肖洪波

佛家说：“凡所有相，皆是虚妄。”

但我们现在却有办法将“虚拟+现实”变得更加真实。

增强现实（Augmented Reality，简称AR）简单来说是通过电脑技术，将虚拟的信息应用到真实世界，真实的环境和虚拟的物体实时地叠加到了同一个画面或空间同时存在。增强现实提供了在一般情况下，不同于人类可以感知的信息。它不仅展现了真实世界的信息,而且将虚拟的信息同时显示出来，两种信息相互补充、叠加。

增强现实有多种定义，为大家广为接受的定义是将虚拟物体添加到现实世界中，为用户提供更为丰富的体验和信息，如同Paul Milgram所定义：通过增强现实技术把计算机生成的图形叠加到真实世界中。



增强现实定义图

自从20世纪70年代早期Pong进入电子游戏厅以来，视频游戏走进我们的生活已经有40多年了。增强现实的新技术，将通过增强我们的所见、所听、所感和所闻，进一步模糊真实世界与计算机所生成的虚拟世界之间的界限。

增强现实在某种意义上已经逐渐在世界上普及开了，应用于市场、游戏和娱乐等各个产业。举个例子，球迷看世界杯的话都知道，球场上经常会出现比如任意球距离的标记，或者某个赞助商的标记。这当然不是直接印在草皮上的，而是电脑合成后加在比赛影像上的。这就是增强现实最基本的应用。特别是作为一种全新的人机交互方式，增强现实技术和应用的发展在当前更加备受关注，而智能手机是当仁不让的增强现实应用的最佳平台。因为在移动互联网时代，人们迫切需要更多的互动性、更多的即时性、更多的个性化、更多的垂直化，随着AR技术与智能手机发展的结合，相信一个可以连接虚拟世界和物理世界的移动互联网上的杀手级应用正在向我们走来。

在AR领域，一些欧洲公司走在了世界的前面，如Layar、Wikitude以及Junaio公司，早在2009年和2010年就发布了在智能手机上应用的增强现实应用程序。用户将手机摄像头指向某物就可看到一幅各种信息重叠显现的图像，如地理方位、大楼名称、历史图片或饭店一览等，甚至还能显示出用户最近在Twitter和Facebook网站上的更新。利用增强现实应用程序打广告的方式也开始流行起来。

市场调查研究公司Juniper Research在其最新

Junaio AR Browser

Layar AR Browser

公布的报告中表示，对移动应用程序中的融合增强现实技术的日益重视将推动这类程序的下载量大增。预计2015年的全球下载量将高达14亿次，而2010年，这一数字仅有1,100万。

Juniper的报告还显示，目前具备增强现实功能的应用程序的数量急剧上升，而现有产品的广度也已经显著扩大——从一开始的位置搜索程序和浏览器扩大到游戏、社交网络、教育、生活方式和个人医疗保健等诸多方面。这些进展的取得大多伴随着具备增强现实功能的智能手机的迅速普及。

此外，该报告还指出，众多品牌对增强现实技术的认识和兴趣在2010年下半年急剧上升，多个知名品牌开始研制带有增强现实内容的应用程序，或者利用现有的移动增强现实应用程序让终端用户了解广告活动中的这类元素。

《移动增强现实报告》中的其他重要发现还包括。

- 预计到2015年，移动增强现实应用程序和服务的收入将接近15亿美元；2010年，这部分收入不到200万美元。
- 预计到2015年，带有增强现实元素的企业应用程序的收入将排在第三位，仅次于位置搜索和游戏。
- 各国政府和移动内容监管机构可能需要修改或更新现有有关隐私、诽谤、版权的法规，将增强现实应用程序纳入其中。

最近在国内发布的触景AR浏览器是目前可利用最多智能手机传感器（如陀螺仪、GPS、摄像头、重力感应器、3G通信）的一项新型手机应用，可以极大地调动用户参与的热情，还原互联网的真实感，让互联网跟真实世界的关系更紧密，是现在LBS应用和图像识别服务的更高融合。AR将为互动营销、移动广告发展带来了巨大的商机，它创造出全新的用户体验，在一种轻松活跃的环境中，使用户与品牌产生零距离的接触。

随着触景AR浏览器和开放API的内容发布平台的推出，我们正在看到一个新的互动式个性化用户体验媒体和应用平台的诞生，这个平台将带来以下改变。

1．**全新的广告形态**：用增强现实技术实现的虚拟广告投放，大大的有别于报纸广告、电视广告、户外广告、搜索引擎广告。

2．**全新的搜索模式**：我们将不再低着头在搜索引擎的输入框输入文字来查询信息；我们将抬起头，把摄像头作为我们感知周边世界的浏览器，通过在不同的图层中切换，找到感兴趣的垂直信息，体验更加丰富的增强现实。

3．**全新的游戏模式**：虚拟的游戏与周边的实景融为一体，而玩家就在真实与虚拟的游戏场景中切换，体验庄周、蝴蝶物化合一的境界。

4．**全新的社交方式**：利用AR技术，未来每个人都可以有一个立体的社交主页和生活流日志，并能与微博和各种SNS集成，随时随地立体地发布和分享内容，甚至通过人脸识别技术，用户就能利用摄像头看到你的信息、动态，添加你为社区好友。

5. **全新的生活方式**：AR将彻底改变人们的生活方式。在没有AR之前，物理世界和虚拟世界是分离的，现实世界和梦幻世界是阻隔的，你的过去、今生和来世是不可相遇的。而AR将再一次放飞我们的想象力，让未来与当下握手，让历史浮现眼前，让梦境与现实融合，让生活充满精彩，我们将要迎来一个全新的移动互联网时代。

如果说“存在就是被感知”，那么我们现在可以通过AR来感知更多的、更精彩和更真实的世界！

“多么蓝的天啊！走过去，你就融化在蓝天里！”——还记得这句小时候看的电影《追捕》中的著名台词吗？现在结合AR的体验，我们不用舍身一跳也可以达到这种“心醉神迷，物我消融”的“至乐”境界。有幸生活在今天的我们居然可以用带有AR功能的智能手机感受王阳明需要通过“格竹”和“看花”而追求的那种“良知独显，与造物者游”的审美体验。

肖洪波

2001年毕业于清华大学，先后供职于HP、CA、IBM，担任技术顾问和架构师等职务。于2010年参与创建触景无限，专注于移动领域的增强现实技术和云计算服务。目前负责公司的技术和数据服务，担任CTO职务。

Android操作系统移植经验分享（下）

文 / 钟文昌

本刊2011年5月期刊登的“Android操作系统移植经验分享”上篇中，作者介绍了Android 移植步骤以及流程、准备工作、Linux内核移植等内容，本文就整合性修改、编译运行、开发环境等方面继续深入分享Android移植的重要经验。

整合性修改

在成功移植Linux内核之后，接着要对Linux内核、Linux 设备驱动、Android 程序库、Android应用及Android运行环境等作整合性的修改。我们必须先下载Android原始代码，这里使用的是cdma-import版本。以下是我们的操作步骤（详细流程可以参考http://source.android.com/download）：

```
curl http://android.git.kernel.org/repo > repo
chmod +x repo
./repo init -u git://android.git.kernel.org/platform/manifest.git -b cdma-import
./repo sync
```

我们在开发过程中遇到三个比较严重的问题包括。

- LCD：画面闪烁。
- 键盘：缺少driver。
- Touch：更换driver。确认设备驱动运行正常之后，不论我们触碰触摸板的任何位置，其总是回复（0，0）这个坐标。

我们将上述问题归纳为设备驱动的修改或重新撰写，以及与Android 程序库、Android application及Android运行环境等有关的问题。修改现有Linux 设备驱动程序代码的部分建议参考同类型且正常运作的设备驱动。下面，我们将详述其解决方法。

LCD

在成功移植Linux内核的情况下，运行Android会看不到LCD的显示画面，因为Android需要Double Framebuffer，所以必须修改Framebuffer驱动以获得两倍Framebuffer的存储器，这部分可以参考Android Emulator所使用的Framebuffer Driver：linux-2.6.25-android-1.0_r1/drivers/video/goldfishfb.c。接着处理不同平台的相应问题，我们遇到LCD闪烁的问题，如图7。LCD闪烁是因为LCD 调节器不断重复地开关所造成。而其不断重复地开关是因为Framebuffer驱动中的LCCR0（暂存器）设定值没有同步，解决方法是增加fb_pan_display函数以更改LCD的状态。

图7 LCD画面闪烁，由左至右每张图像拍摄的时间间隔为1秒

键盘

我们的目标平台缺少键盘驱动程序，解决方法为新增目标平台的键盘驱动程序（参考linux-2.6.25-android-1.0_r1/drivers/input/keyboard/pxa27x_keypad.c，并作大幅度修改）。

由于我们的目标平台仅有4x4 matrix buttons，因此我们使用组合键的方式来扩增按键数，但按键数还是不够，所以我们只选取某些特定或重要的按键值，而忽略部分按键值。此外，因为该平台的键盘没有提供硬件中断，

我们使用polling的方式来解决这个问题。考虑到低功耗及电源管理在移动装置上是一项值得探讨的问题，因此我们实际采用了一个有效的算法来检测键盘被按压的情形，并适当地降低polling频率以达到降低功耗及省电的效果。

如何在键盘被按压时能够实时反应，而且能在没有被按压的情况下降低polling的频率以减少耗电量？基于按压的行为也具备有时间区域性（Temporal Locality）现象，我们使用一个简单的算法：当键盘被按压时，将polling的时间间隔设定成预设的最小间隔，使能在被按压的期间集中检测，但是如果在间隔时间内未被按压，我们将逐步增加polling的时间间隔（interval）直到我们预设的最大时间间隔，如此便能在被按压可能性较小的时候达到省电的效果。

Touch Panel

因为触摸屏驱动程序在新版Linux内核作了很大的改动，所以要先确认新版的触摸屏驱动程序可以正常运作，接着观察其在Android的环境下是否有其他问题。我们使用HTC Dream作为相关问题的对照组。我们修正触摸屏驱动程序的四个问题：

指定IRQ、Connect to Input Subsystem、校正从硬件读取的（X，Y）坐标值、调整触碰的灵敏度。

使用printk验证触摸屏驱动程序所得到的坐标值，在此需要知道一个touch事件会包含：X坐标值、Y坐标值、与EV_SYN[可参考Linux内核原始代码linux/include/linux/input.h。]（表示完成一个包含X、Y坐标值的touch事件）。在触摸屏驱动程序运作正常的情况下，Android application对touch的反应是不论我们触碰touch的任何位置，所得到的坐标值似乎总是（0，0），于是我们追踪touch事件在Android的运作流程，并以下列确认的方式逐步发掘该问题的成因。

1.触摸屏驱动程序是否正确将（X，Y）回报至Android application?

2.Android application是否收到正确的（X，Y）？

3.若是2成立，是否因为某些条件造成（X，Y）被改变?

图8为在Android环境下，输入装置与电源管理的关系。我们观察Android运行时的消息确认Android可以取得touch的相关信息，包含（X，Y）的最大及最小值等，但不知道在流程中的哪个环节发生问题，导致不论触碰touch的任何位置，Android的画面总是反应触碰到（0，0）。Android的source code约2GB且没有UML Diagram，因此我们采用暴力法（需要熟悉Linux及Shell的操作指令，包含find、grep、|（pipe）的用法等等）及ctags（我们使用ctags --C++-kinds=+p –R来追踪Android原始码）并依靠经验来追踪Android原始代码。

我们先从Android运行时的消息中找到与touch相关的信息，根据其信息以frameworks/base/services/java/com/android/server/KeyInputQueue.java为追踪Android原始代码的切入点。我们发现Android application会判断当前LCD的状态（ON/OFF）来决定是否处理touch事件（LCD关闭时不需要处理touch事件），而LCD的状态又相依于Battery的状态（电量及是否外接电源等）。因为我们的目标平台没有Battery device及Battery driver，所以Android无法得到Battery的相关信息，以致认为LCD关闭着而忽略触摸屏驱动程序回报的（X，Y）。紧接着收到触摸屏驱动程序的SYN事件后，便完成

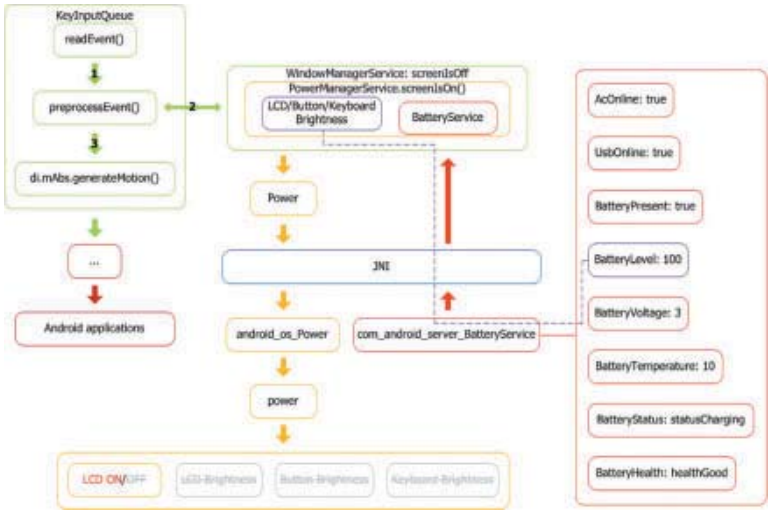


图8　在Android环境下，输入装置与电源管理的关系

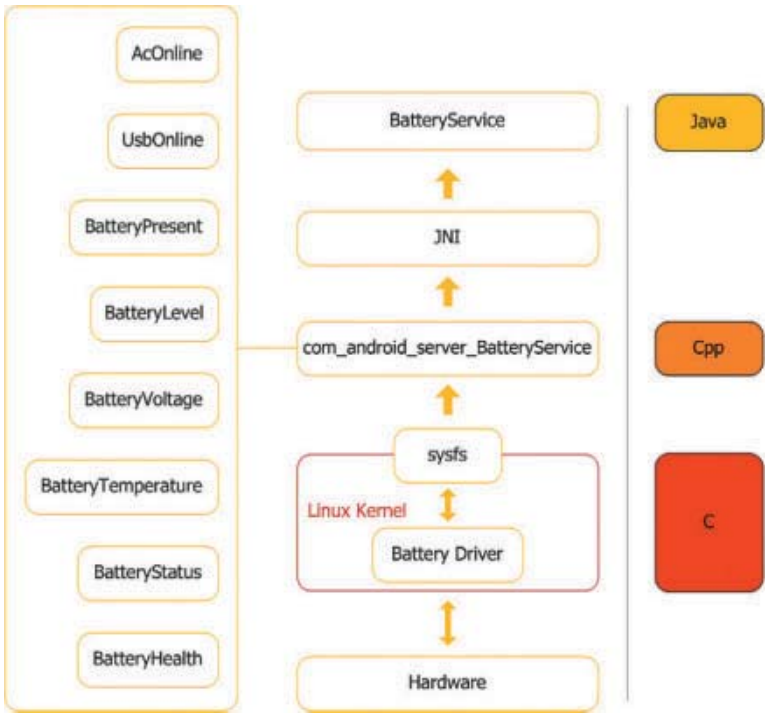


图9 Android的BatteryService流程

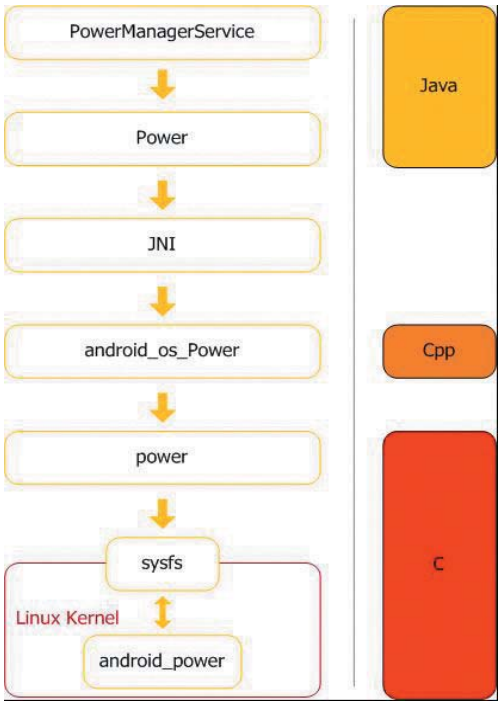


图10 Android的PowerManagerService流程

一个touch事件。由于程序中（X，Y）的预设值为（0，0），所以Android误认为当下的（X，Y）为（0，0），以致不论触碰touch的任何位置，Android的反应都是（0，0）。

根据这样的追踪结果，我们修改程序代码以回报假的Battery状态，固定LCD处于被点亮的状态，并使得Android得到良好的电源及电池状态。此外，由于我们的LCM不支持背光调整，所以我们只许LCD被点亮，而不提供亮度调整的功能。因此我们修改hardware/libhardware/power/power.c，并更改相关文件（Android通过相关文件来控制硬件的亮暗及背光）的存取权限，使得touch可以正常运作。图8为经过我们修改之后，在Android环境下，输入装置与电源管理的关系与设定值，图9为Android的BatteryService流程，图10为Android的PowerManagerService流程，图9及图10为图8的详细流程。

编译与运行Android

在完成整合性修改之后，我们需要重新编译Linux内核及Android，因为Android不支持JDK6，所以我们使用JDK5（http://java.sun.com/javase/downloads/index_jdk5.jsp）。在Android编译完成后，我们需要取用system及data这两个目录来运行Android。因为我们的目标平台不是G1等HTC手机，所以不管是MTD partition的规划、/system及/data的挂载位置可能皆与Android原先的设置不同；再者为了使touch可以正常运行，我们必须修改部分文件的存取权限。最后因为我们的目标平台没有蓝牙装置，所以我们将部分与蓝牙相关的设置移除。init.rc在经过以上修改之后，Android才能够在我们的目标平台上顺利地运行。

在成功编译Linux内核及Android之后，我们将kernel image写入Flash ROM并确认Target可以正常开机。因为我们的目标平台PXA270仅有32MB Flash ROM，不足以存放Android的文件系统，所以我们使用U盘建立一个较为完整的文件系统，并使用chroot于系统启动之后，由存在于Flash ROM的基础文件系统转换至USB上的Android文件系统并运行之。在USB的文件系统中，我们加入Android成功编译后的system及

data两个目录；这两个目录的所在的路径为out/target/product/generic/system及out/target/product/generic/data，复制这两个目录至U盘中，图11为Android在我们目标平台PXA270上的目录结构。最后修改init.rc以符合我们实验平台的环境。

图11　Android在我们目标平台PXA270上的目录结构

若是在运行Android的过程中遇到问题，可以使用logcat及strace来调试。我们的目标平台PXA270在Android环境下，经过连续7天不关机的测试，仍然可以正常操作，表示我们的目标平台在Android下具备一定的稳定性，证明我们的移植是成功的。我们录制Android的操作影片，并上传至Youtube，图12为Android在我们目标平台PXA270的操作影片截图。

图12　Android在目标平台PXA270的操作影片

开发环境

要建构一个好的building环境有一定的困难度，因为要考虑到实用性、跨平台、平行化编译、编译顺序、重复编译等问题，所以一般building环境都是为了特定目的而开发，并具备一定客制化的程度。

传统Building 环境与Android Building环境

传统building环境以Makefile来控制编译的行为，而Android的开发环境是以Android.mk来取代Makefile的角色，如图13所示。

若以树形图表示开发环境的目录结构，则传统开发环境除了最末端以外的每个层次中最少会有一个Makefile，由根目录的Makefile以递归方式呼叫下一个层次的Makefile，通过此方式完成整个编译程序。此架构的缺点是各层次之间的相依性太强，例如某层次的Makefile可能参考到上层甚至是根目录Makefile中的参数等，造成单一Makefile可能影响的纵深非常深，衍生building环境日后不易维护且通常无法独立编译某层次的某个控件等相关问题。

Android的开发环境为一整合型的开发环境，它希望此环境下的套件能够被独立地编译，如此增加平行化编译的可行性，于是它改写其中某些套件的编译方式，希望它们能够被独立地编译且减少相依性。此环境会将编译时使用到的变量声明、路径位置、函数定义全部收集在build/core下，并使用各自的Android.mk来控制各套件的编译行为。Android开发环境的优点是在了解其规则的情况下，开发人员可以很方便地将自己的程序代码加入Android中，并在相对应的目录下加入Android.mk，而Android的开发环境会通过Android.mk来编译相关的程序代码；但缺点是Android的开发环境为了达到这种后期开发的便利性，牺牲了部分编译时间，牺牲的时间是耗费在寻找相关的Android.mk。为什么Android的开发环境有办法聪明地找到开发人员新增的Android.mk，而编译新增的程序代码？因为在Android的开发环境中使用大量Linux的find指令，因此会对储存装置造成较大的负担，所以开发Android要尽可能准备高速或是分散式的储存装置，并搭配高阶或是多核的CPU来加快开发速度。

我们的开发环境

我们为此打造一个全新、整合性的building环境，我们的building环境不仅小而美，还有下列优点：

1.分散式且集中管理的building环境。

- 我们将自己建立的Makefile收集在mkfile目录下集中管理，避免传统recursive make的Makefile散布在不同层次的目录中造成管理不易等问题。
- 单一Makefile只负责单一控件，避免传统使用单一Makefile却要控制所有控件的编译行为而造成Makefile日益增大及维护不易等问题。

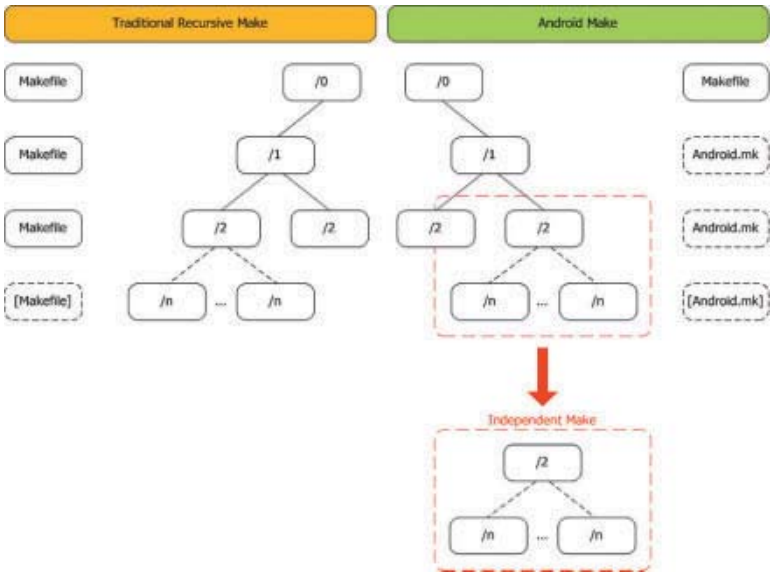


图13 传统recursive make与Android 开发环境的比较

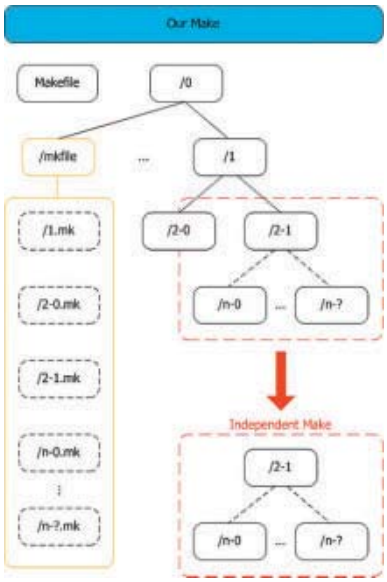


图14 我们的building环境

■ 降低Makefile之间的相依性，building环境易于维护。

2.尽可能避免recursive make，可在GNU Make支持parallel make的情况下，同时编译多个目标、缩短编译时间、加快开发过程。

3.结构化的building环境。

■ 控件（Component）化的building环境。

■ 新增的控件可以非常容易地加入我们的building环境中。

4.在Makefile中使用函数呼叫取代变量定义，减少变量声明。

5.加入判断式以避免重复编译等问题。

我们设计此开发环境的目的是希望使用者可以轻松地通过make指令便完成Android运行环境的建置，所产生的Binary Code可以直接写入Flash ROM，并复制目标平台的文件系统至U盘中，如此目标平台开机后便可以看到Android的运行画面。此外，我们的开发环境亦支持以往Embedded Linux的开发方式，我们将Android的文件系统独立置于U盘中，由使用者自由选择是否要运行Android。

为进一步了解我们的building环境的编译效能，我们在IBM X260服务器上进行了简单的测试，表2为测试环境与结果列表。该服务器包含两颗双核心的Xeon，因此具备四个CPU核心。由结果我们可知make需要1小时零28分钟来完成编译的工作，当使用-j参数指定可同时运行的job之后，时间可获得大幅的改善。

表2 我们的开发环境在不同平台上的测试数据

Operating System	CentOS release 5.3 (Final)
Linux内核	2.6.18-128.1.10.el5，x86_64
CPU	4x Intel(R) Xeon(TM) MP CPU 3.16GHz
DRAM	4GB
GNU Make Version	3.81
Make Command	Time (hh:mm:ss)
make	1：28：39
make –j2	53：51
make –j4	47：37

结论

本文以移植的流程为主轴，使用引导的方式，从大方向到小细节、从概念、流程到相关程序，尽可能在有限篇幅中详述Android移植的过程。我们修改程序代码的原则是尽可能以可读性最高及最少的修改来达到我们预期的目的。从系统整合的角度来看，要完成Android系统整合的工作并非易事，所以我们希望公开移植的流程，分享移植的经验，在不侵权的前提下，尽可能公开移植代码，这也正是开源及Android的理念。

钟文昌（Mask）

台湾第一位并且也是开设最多Android Porting系列课程的业界讲师，提供Android企业培训及顾问服务。博客地址为：www.mask.org.tw/blog。

责任编辑：常政（changzheng@csdn.net）

高校“战队”涌现

TCL Android TV开发大赛进入冲刺期

自6月TCL Android TV开发者大赛“校园宣讲团”走进高校之后，在北方工业大学、天津工业大学等高等院校，由热爱程序开发的学子们成立“Android战队”正成为一种热潮。

伴随着各高校“Android战队”宣布加入角逐，由TCL发起的这场持续长达四个月之久的智能电视程序开发大赛，一时间再起波澜，大赛提前进入到冲刺期。

多队高校“集团军”加入大赛角逐

“TV-Game研发团队毕业学员杨二峰带领的TCL战队组建中，欢迎北方工业大学编程爱好者赶紧加入他的团队，他有专业的TV-Game研发经验，也欢迎外校人员客串加盟。”在TV-Game论坛的“TCL大赛直通车”专区，记者看到，帖主“snowlf”在论坛中发出了“英雄邀请函”，正为组建参赛战队而四处邀约自己学校的程序开发达人。在国内知名的V城社区，除了有帖主“snowlf”所提到的杨二峰团队之外，还有天津师范大学、北京航空航天大学、北京邮电大学等高校多支“战队”。

值得关注的是，这次大赛还引来了创业团队“第五发明”和号称V城社区天将神兵的“Air Stone”团队的参与。“Air Stone”团队此前以专为企业与应聘者提供第三方咨询服务平台的开发及运营而在Android开发领域已经小有名气。对此，有参赛者预测，“Air Stone”团队很有可能会在此次将其在Android领域的随身招聘客户端开发成果移植到TCL的Android系统上来，让未来的智能电视也新添随身招聘求职，把电视转变为便利的招聘或求职工具。面对强敌，天津师范大学的“第五发明”战队队长王懿也不失信心，称将积极组队并与搭档“玛雅风神”定好参赛项目之后，全力备战此次大赛，将直指TCL大赛桂冠。

家庭益智类游戏开发占优势

TCL Android TV开发者大赛由在智能消费电子领域领先的全球知名企业TCL于4月1日发起。在长达四个月的赛程里，TCL将在CSDN、机锋论坛、安卓论坛、eoe Android 论坛等Android技术专业网络平台开设大赛专区，招募专业人士参与大赛，并给予重金奖励获胜者。目前，在安卓论坛、机锋论坛等大赛协办论坛里，已经有近百件基于Android的电视功能创意作品提前面世，功能涵盖射击游戏、家庭互联等。各参赛选手所开发出的各种趣味功能，从家庭游戏娱乐到商务应用开发等，将Android系统所具有的开放性平台应用发挥到了极致。

据记者初步统计，在当前所提交的参赛作品当中，大部分参赛选手选择的是以家庭益智类游戏的Android开发为主。网友“bile13”认为电视的功能程序开发，应该要以游戏类或娱乐性较强的功能为主，因为电视的属性就是用来家庭娱乐和放松的，并表示“儿童益智类的更好”。

更为欣喜的是，6月30日，全球彩电巨头TCL还联合派拉蒙、孩之宝以“TCL：和变形金刚一起3D智变”为主题，共同在广州的太古仓码头举办了一场盛大的全球刚丝3D狂欢派对。作为《变形金刚3》彩电业唯一国际联合推广伙伴，TCL在派对现场运用酷炫的AR技术（即增强现实技术），向全球发布了首款《变形金刚3》纪念版电视智变系列和国内首个智能3D电视卡通形象——电视金刚“小智”，以推动智能3D的全面普及。

主持人：张银奎
《软件调试》一书作者，从事软件开发和研究10余年，对IA-32架构、操作系统内核、虚拟技术，尤其对软件调试有较深入的研究。翻译（合译）作品包括《数据挖掘原理》、《机器学习》、《人工智能：复杂问题求解的结构和策略》、《观止——微软创建NT和未来的夺命狂奔》等。

混乱数据何处来

标准文件流有关的陷阱

软件问题千差万别，书本上介绍的东西往往都是简化过的，要比现实软件的现实问题简单很多。因此我一直执拗的坚持做开发工作，不脱离代码，保持手中至少有一个需要编码的工作项目。软件的根本在代码，智慧和糟粕都在其中，能看清代码便看到了软件的关键。看代码有很多种境界，能看懂源代码是一层境界，没有源代码时能看懂汇编又是一种境界，源代码和汇编都没有，从软件的行为感知背后的代码再是一种境界。软件工程师要修炼的第一手功夫就是对代码的感知力和控制力。说得有点绝对，一家之言，谨供参考。言归正传，今天仍然介绍一个的真实的问题，来自软件开发第一线实际问题，让人感觉见了鬼的问题，找到答案后，发觉也是一个有源代码可查的问题，但是找到有关的代码却不是一件容易的事情，而且当找到代码后，仍然很难说清是哪段代码写错了。或者说这不是一方可以促成的问题，而是多方“努力”的结果，单独看每一方都有情有理，但加在一起，却铸成大错，真的很“纠结”。让我们从头说起吧。

用虚拟文件通信

问题发生在Linux平台上，涉及到两个模块，一个是运行在用户态的后台程序，即通常所谓的Daemon，相当于Windows中的系统服务，另一个模块是运行在内核态的驱动程序。这两个模块大多数时候各做各的，但偶尔需要沟通一下，沟通的方式是虚拟文件，也就是通常所谓的Sysfs。Sysfs又叫设备驱动程序文件系统，是一种虚拟的使用内存做存储媒介的特殊文件系统，其主要目标就是用来与驱动程序通信，可以用读写的文件方式来与驱动程序交换数据。举例来说，可以用来观察驱动程序的状态，也可以用来配置驱动程序。进一步说，驱动程序可以使用内核函数sysfs_create_file()、sysfs_create_bin_file()或者sysfs_create_group()来创建虚拟文件，调用以参数形式传递一个device_attribute结构，在数据结构中指定好虚拟文件的名称、文件模式（读/写共享方式）以及当有人读写这个文件时将被调用的回调函数。

```
struct device_attribute {
struct attribute attr;
ssize_t (_show)(struct device _dev,
char _buf);
ssize_t (_store)(struct device _dev,
const char _buf, size_t count);
};
```

对驱动程序来说，每个虚拟文件就是一个设备属性，虚拟文件所在的目录对应的内核对象，虚拟文件系统（sysfs）会管理这个映射关系。

图1画出了问题涉及的两个模块的简单架构图，上面是运行在用户态的后台程序，不妨简称它为MyDaemon，下面是运行在内核态的驱动程序，简称MyDriver，二者通过

图1 使用虚拟文件与驱动程序通信

驱动程序创建的虚拟文件/sys/mydevice/vfile来通信。通信的内容其实非常简单，就是一个整数，整数的范围为0到100，MyDaemon定期将这样的整数像写文件一样写下去，为了方便驱动程序接收和转化，MyDaemon应该把数字转换为使用十六进制数表达的字符串，比如数字100应该写为“64”两个字符再加一个0结尾，共三个字节。根据这样的接口规范，用户态MyDaemon中的有关代码如下：

```
        int     fd_sysfs_vfile = -1;
        // 在主函数中打开虚拟文件
        fd_sysfs_vfile=open("/sys/
mydevice/vfile", O_BINARY|O_RDWR);

        int send_to_driver()
        {
            int p,d,ret;
            char hex[3];
            if(fd_sysfs_vfile==-1)
                return -1;

            p = g_last_result*100.0;
            d = (p/16);
            hex[0] = d<10?(d+48):(d+55);

            d = (p%16);
            hex[1] = d<10?(d+48):(d+55);
            hex[2] = 0;
            lseek(fd_sysfs_vfile, 0, SEEK_
SET);
              ret = write(fd_sysfs_
vfile,hex,3);
            if(ret<0)
            {
               d4d(LOG_FATAL, "write sysfs
failed with %d\n", ret);
            }
            // flush(fd_sysfs_vfile);

            return ret;
        }
```

上面函数中为hex数组赋值的那部分可以替换为调用sprintf()，这样写只是为了减少一些开销，其中的d4d子函数是用来打印调试信息的，d4d是Design for Debug的缩写，这样说明之后，大家看出有什么不妥的地方吗？

写了没有反应

两个模块都写好了后，放在一起集成，很快发觉这个使用虚拟文件的通信机制有问题，使用cat命令观察虚拟文件的内容，值始终不变，总是64，按照约定的语义解释，也就是100。

首先考虑到的是检查后台程序中的通信函数，也就是上面列出的send_to_driver()函数。结果无论是用打调试新的方法，还是使用gdb跟踪，这个函数工作的都非常好，文件描述符有效，转16进制数也很正确，lseek也正常，然后看着write函数将转好的结果写下去，返回值为3，一点问题没有，每一步都执行得很顺利。但当使用cat命令观察虚拟文件时结果就是不变，始终是64。“写了没有反应，怪啊!”尝试增加调用flush函数，也没有任何效果。实际上，因为虚拟文件就是基于内存的虚拟文件系统，所以flush函数没有什么意义。

接下来考虑的当然是检查驱动中的代码，通信是双方的事，一方检查好了，当然还要检查另一方。不过驱动中的代码也简单的出奇，负责处理写虚拟文件的回调函数只那么三四行代码，把系统传进来的字符串转换为数字，然后就用条件运算符赋值给全局变量了。

```
static ssize_t
vfile_store(struct kobject *kobj,
struct kobj_attribute *attr,
                        c   o   n   s   t
char *buf, size_t n)
{
        unsigned long m = 0;

        m   =   simple_strtoul(buf,
NULL, 0x10);
g_vfile_pecent  =  (m  >  100)  ?  100:
m;

        return n;
}
```

这么简单的几行代码有什么问题吗？如果说有，那就是缺少一行支持调试的打印调试信的语句，比如printk之类的语句。打印调试信息在Linux中仍是比较常用的调试手段，特别是内核态，因为Linux的内核调试不如Windows那样方便。

看了用户态和内核态的相关代码后，可以尝试猜测一下是什么样的问题。至少有这几种可能：

用户态程序send_to_driver函数的写文件动作没有正确触发内核中的回调函数vfile_store，vfile_store可能根本没有被调用，所以写了也没有用，不知道写到哪里了。

用户态的写动作顺利触发了内核中的回调函数，但是传下来的数据出意外了，内核函数收到后，转换出来总是大于100，结果总是把

100赋给全局变量。

到底是哪种情况呢？空想到这个时候可以了，接下来需要“实地考察”了，那就加print语句吧。

超大数字何处来

在Linux做开发，如何打印调试信息是种很重要也很讲究的技术，写好print语句不是件很简单的事。对于本例，先在函数的return语句上面加入了这样一句：

```
printk(KERN_WARNING "vfile_store got value %d\n", m);
```

其中KERN_WARNING是一个字符串常量，被定义为“<4>”，用来代表信息的重要级别。

增加以上语句后，重新编译驱动，更新到目标系统，重新启动，再和用户态的后台程序集成，使用dmesg | grep vfile寻找新增加的调试信息。

首先，真的找到了，看来上面的猜测A是不成立的，驱动中的回调函数确实被调用了。但是仔细看n的取值，不禁大叫一声：“怎么这么大的值啊？”

```
vfile_store got value 50010415
```

看到这么大的数字后，前面的猜测B有点靠谱的。但是反复查看后台程序的send_to_driver函数，它不可能写下来这么大的数字啊？因为一共就写两个字符下来，也就是只有两个十六进制位啊。

有位在Linux下做了很多年开发的朋友曾经对我说，在Linux下调试问题，主要靠想，努力地想，想出线索来，再加print。的确还需要继续想一想，基于目前的证据，前面说的情况B还有可能，但不能确认，此外，可能还有一种情况：

除了我们看到的send_to_driver函数外，还有别的地方在写这个虚拟文件，这个奇妙的超大数字是来自我们没有注意到的其他地方。

如果有交互式内核调试，那么可以在vfile_store函数设置一个断点，当有人写的时候当即拿下，中断下来，看到底是谁在写这个文件，写的数据到底又是什么内容。可是，内核调试环境不容易建立啊。还得靠printk，要把刚才那句调整下，多打印出一些信息，索性将回调时传进来的字符串都打印出来，于是改成：

```
printk(KERN_WARNING "vfile_store got buffer %s, n=%d, m=%d\n", buf, n, m);
```

修改以后，再次重新编译驱动，更新到目标系统，重新启动，再和用户态的后台程序集成，使用dmesg | grep vfile寻找新增加的调试信息，看看这下是否有新线索。

这次的结果出乎预料，但内容似曾相似，仔细核对一下，新打印出来的字符串居然是用户态程序中的调试信息，也就是printf函数中指定的字符串参数。因为字符串很长，开头的一部分恰好又是按“年月日小时分秒”格式输出的时间串，这样的串被驱动程序接收到后，转换为16进制，可不是很大的数字么！

分析到这里，基本清楚了如下几点：

1.后台程序写数据下去没有反应，是因为写的内容被后面误写的大数字给覆盖了，因为后台程序中刚好每次向驱动写数字后便有调用printf输出信息的语句。

2.意外的大数字来自后台程序中的printf调用。

按理说，后台程序中不应该再调用printf，之所以代码中仍存在调用printf的语句是因为这个后台程序有两种运行模式，一种是以普通的控制台方式运行，便于调试和测试，另一种是以后台（daemon）方式运行。另一个原因就是没有预料到printf会导致这种效果。

两流为何入一渠

接下来的重点是为什么调用printf打印的调试信息会打印到用来与驱动程序通信的虚拟文件中？虚拟文件和调试信息输出应该是两个截然不同的路径，二者应该各走各的路线，现在为什么调试信息流入到虚拟文件中了呢？这好比是两股本该在两条独立管道流淌的液体突然混合到一个管道中。

不妨梳理一下这两个管道的来龙去脉。先看虚拟文件，后台程序调用open函数打开虚拟文件，把返回的文件描述符保存在全局变量

中，写文件时使用这个全局变量。

```
fd_sysfs_vfile=open("/sys/mydevice/
vfile", O_BINARY|O_RDWR);
```

再看printf函数，根据函数的规约“Print formatted data to stdout”，这个函数应该把信息打印到stdout，也就是所谓的标准输出设备。查看GNU libc的源代码，可以看到这个函数的典型实现：

```
        // 来自glibc-2.8-20090518 printf.c
        int
        __printf (const char *format,
...)
        {
          va_list arg;
          int done;

          va_start (arg, format);
          done = vfprintf (stdout, format,
arg);
          va_end (arg);

          return done;
        }
```

使用的也是全局变量，相关的全局变量有三个，即所谓的三个标准流，标准输入、标准输出和标准的错误输出：

```
        // 来自glibc-2.8-20090518 stdio.c
        _IO_FILE *stdin = (FILE *) &_
IO_2_1_stdin_;
        _IO_FILE *stdout = (FILE *) &_
IO_2_1_stdout_;
        _IO_FILE *stderr = (FILE *) &_
IO_2_1_stderr_;

        //来自glibc-2.8-20090518 stdfiles.
c
        DEF_STDFILE(_IO_2_1_stdin_, 0, 0,
_IO_NO_WRITES);
        DEF_STDFILE(_IO_2_1_stdout_, 1,
&_IO_2_1_stdin_, _IO_NO_READS);
        DEF_STDFILE(_IO_2_1_stderr_, 2,
&_IO_2_1_stdout_, _IO_NO_READS+_IO_
UNBUFFERED);
```

简单来说，上面的定义就是定义了三个文件指针，这三个指针指向三个全局结构，这三个全局结构中的文件描述符字段固定为0、1、2这三个值。也就是说三个标准流固定对应到了0、1、2这3个文件描述符，在头文件unistd.h中，也可以看到这一点：

```
        /* Standard file descriptors.  */
        #define   STDIN_FILENO     0
/* Standard input.  */
        #define   STDOUT_FILENO    1
/* Standard output.  */
        #define   STDERR_FILENO    2
/* Standard error output.  */
```

这样分析之后，可能的情况是两个函数使用的文件描述符意外相同了，所以导致两个数据流混合到一个流中了。怎么导致的两个文件描述符相同呢?

中肯的建议

关于如何写后台程序，有很多很好的指导文章，不仅讲解的细致周全，而且还有示例代码供参考。在这些指导文章中，详细列出了作为一个后台程序，应该如何做哪些工作，先做什么，再做什么。最后一步，通常都有这样一个建议：关闭标准文件描述符。也就是执行类似下面这样的代码：

```
        close(STDIN_FILENO);
        close(STDOUT_FILENO);
        close(STDERR_FILENO);
```

也有建议写一个小循环的，也就是：

```
        for(i=0;i<=STDERR_
FILENO;i++)
          close(i);
```

执行这一动作的理由也非常的充分：既然后台程序不能使用终端，保持这些文件描述符是多余的，而且可能导致安全方面的风险。

不是结论的结论

大家可能已经猜测出了答案，是的，导致问题的原因就是MyDaemon中关闭了三个标准流，而后当它继续执行打开其它文件时，系统复用空闲出来的标准文件描述符，将它们返回给了应用程序，于是虚拟文件的描述符复用了标准输出的描述符，也就是1。这样以来，当写虚拟文件时，使用的是1这个描述符，可以把信息写下去，当执行printf时，仍按printf的逻辑找stdout指针，读取固定的描述符1，然后写下去，岂不知此1非彼1，意外写到虚拟文件中去了。听起来有点滑稽，但实际情况就是这样，把关闭标准流的代码注释掉问题就解决了。

其实不一定非与驱动通信才有问题，只要关闭了标准流，再打开普通文件，那么便有可能出现混乱了。笔者特意写了一个小程序来验证，果然可以重现，在试过的几个Linux平台上，都可以重现。移植到Windows中，居然也可以重现。这也正常，因为库函数的实现都是按照规约去做的啊。

责任编辑：高松（gaosong@csdn.net）

本期问题：

上期答案是：关于关键区结构的SpinCount字段的含义，简单来说，当有线程调用EnterCriticalSection函数希望进入关键区而关键区又忙时，如果SpinCount不为0，那么EnterCriticalSection函数会做循环等待，每循环一次，看一下关键区是否已经空闲，如果空闲便进入关键区，如果不空闲便继续等待，直到循环了SpinCount指定的次数。应用可以调用InitializeCriticalSectionAndSpinCount在初始化关键区时指定SpinCount值。

本期的问题是：文件描述符和文件句柄是一回事吗?

编者说明：

- 投稿信箱：contest@csdn.net
- 联系方式写在一个单独的TXT文件里，包括以下几项：
  1）姓名
  2）工作单位或学校
  3）电话联系方式
  4）邮寄地址
  5）E-mail地址
- 解答提交时间，最好早于当月15日。

并行编程中的“锁”难题

文 / 陈冠诚

在并行程序中，锁的使用主要会引发两类难题：一类是诸如死锁、活锁等引起的多线程Bug；另一类是由锁竞争引起的性能瓶颈。本文将介绍并行编程中因为锁引发的这两类难题及其解决方案。

用锁来防止数据竞跑

在进行并行编程时，我们常常需要使用锁来保护共享变量，以防止多个线程同时对该变量进行更新时产生数据竞跑（Data Race）。所谓数据竞跑，是指当两个（或多个）线程同时对某个共享变量进行操作，且这些操作中至少有一个是写操作时所造成的程序错误。例1中的两个线程可能同时执行“counter++”从而产生数据竞跑，造成counter最终值为1（而不是正确值2）。

例1：

```
#include <pthread.h>
int counter = 0;
void *func(void *params)
{
        counter++; //数据竞跑
}
void main()
{
        pthread_t thread1, thread2;
        pthread_create(&thread1, 0, func,
0);
        pthread_create(&thread2, 0, func,
0);
        pthread_join(thread1, 0 );
        pthread_join(thread2, 0 );
}
```

这是因为counter++本身是由三条汇编指令构成的（从主存中将counter的值读到寄存器中；对寄存器进行加1操作；将寄存器中的新值写回主存），所以例1中的两个线程可能按如下交错顺序执行，导致counter的最终值为1：

例2：

```
load [%counter], rax; // 线程1从counter读
取0到寄存器rax
add rax, 1; // 线程1对寄存器rax进行加1
load [%counter], rbx; // 线程2从counter读
取0到寄存器rbx
store rax [%counter]; // 线程1把1写入
counter的主存地址
add rbx, 1; // 线程2对寄存器rbx进行加1
store rbx, [%counter]; // 线程2把1写入
counter的主存地址
```

为了防止例1中的数据竞跑现象，我们可以使用锁来保证每个线程对counter++操作的独占访问（即保证该操作是原子的）。在例3的程序中，我们使用mutex锁将counter++操作放入临界区中，这样同一时刻只有获取锁的线程能访问该临界区，保证了counter++的原子性：即只有在线程1执行完counter++的三条指令之后线程2才能执行counter++操作，保证了counter的最终值必定为2。

例3：

```
#include <pthread.h>
int counter = 0;
pthread_mutex_t mutex;
void *func(void *params)
{
pthread_mutex_lock(&mutex);
counter++; //处于临界区，不会产生数据
竞跑
pthread_mutex_unlock(&mutex);

}
void main()
{
pthread_t thread1, thread2;
pthread_mutex_init(&mutex);
pthread_create(&thread1, 0, func,
0);
pthread_create(&thread2, 0, func,
0);
pthread_join(thread1, 0 );
pthread_join(thread2, 0 );
pthread_mutex_destroy(&mutex);
}
```

死锁和活锁

然而，锁的使用非常容易导致多线程Bug，最常见的莫过于死锁和活锁。从原理上讲，死锁的产生是由于两个（或多个）线程在试图获取正被其他线程占有的资源时造成的线

程停滞。在下例中，假设线程1在获取mutex_a锁之后正在尝试获取mutex_b锁，而线程2此时已经获取了mutex_b锁并正在尝试获取mutex_a锁，两个线程就会因为获取不到自己想要的资源、且自己正占有着对方想要的资源而停滞，从而产生死锁。

例4：

```
// 线程 1
        // 线程 2
void func1()
        void func2()
{
        {
        LOCK(&mutex_a);
         LOCK(&mutex_b);
        LOCK(&mutex_b);//线程1停滞在此
         LOCK(&mutex_a);//线程2停滞在此
        counter++;
                  counter++;
        UNLOCK(&mutex_b);
                  UNLOCK(&mutex_a);
        UNLOCK(&mutex_a);
                  UNLOCK(&mutex_b);
}
        }
```

例4中的死锁其实是最简单的情形，在实际的程序中，死锁往往发生在复杂的函数调用过程中。在下面这个例子中，线程1在func1()中获取了mutex_a锁，之后调用func_call1()并在其函数体中尝试获取mutex_b锁；与此同时线程2在func2()中获取了mutex_b锁之后再在func_call2()中尝试获取mutex_a锁从而造成死锁。可以想象，随着程序复杂度的增加，想要正确的检测出死锁会变得越来越困难。

例5：

```
// 线程 1
// 线程 2
void func1()
void func2()
{
{
        LOCK(&mutex_a);
 LOCK(&mutex_b);
        ...
         ...
        func_call1();
 func_call2()
        UNLOCK(&mutex_a);
         UNLOCK(&mutex_b);
}
}

func_call1()
func_call2()
{
{
   LOCK(&mutex_b);
 LOCK(&mutex_a);
   ...
   ...
   UNLOCK(&mutex_b);
 UNLOCK(&mutex_b);
   ...
...
}
}
```

其实避免死锁的方法非常简单，其基本原则就是保证各个线程加锁操作的执行顺序是全局一致的。例如，如果上例中的线程1和线程2都是先对mutex_a加锁再对mutex_b进行加锁就不会产生死锁了。在实际的软件开发中，除了严格遵守相同加锁顺序的原则防止死锁之外，我们还可以使用RAAI（Resource Acquisition Is Initialization，即“资源获取即初始化”）的手段来封装加锁解锁操作，从而帮助减少死锁的发生。

除死锁外，多个线程的加锁、解锁操作还可能造成活锁。在下例中，程序员为了防止死锁的产生而做了如下处理：当线程1在获取了mutex_a锁之后再尝试获取mutex_b时，线程1通过调用一个非阻塞的加锁操作（类似pthread_mutex_trylock）来尝试进行获得mutex_b：如果线程1成功获得mutex_b，则trylock()加锁成功并返回true，如果失败则返回false。线程2也使用了类似的方法来保证不会出现死锁。不幸的是，这种方法虽然防止了死锁的产生，却可能造成活锁。例如，在线程1获得mutex_a锁之后尝试获取mutex_b失败，则线程1会释放mutex_a并进入下一次while循环；如果此时线程2在线程1进行TRYLOCK(&mutex_b)的同时执行TRYLOCK(&mutex_a)，那么线程2也会获取mutex_a失败，并接着释放mutex_b及进入下一次while循环；如此反复，两个线程都可能在较长时间内不停的进行“获得一把锁、尝试获取另一把锁失败、再解锁之前已获得的锁”的循环，从而产生活锁现象。当然，在实际情况中，因为多个线程之间调度的不确定性，最终必定会有一个线程能同时获得两个锁，从而结束活锁。尽管如此，活锁现象确实会产生不必要的性能延迟，所以需要大家格外注意。

例6：

```
// 线程 1
// 线程 2
void func1()
void func2()
```

```
{
        {
        int done = 0;
         int done = 0;
        while(!done) {
         while(!done) {
        LOCK(&mutex_a);
         LOCK(&mutex_b);
        if (TRYLOCK(&mutex_b)) {
           if (TRYLOCK(&mutex_a)) {
        counter++;
             counter++;
        UNLOCK(&mutex_b);
             UNLOCK(&mutex_a);
        UNLOCK(&mutex_a);
             UNLOCK(&mutex_b);
        done = 1;
                       done = 1;
      }
           }
      else {
          else {
        UNLOCK(&mutex_a);
      UNLOCK(&mutex_b);
      }
           }
    }
        }
}
        }
```

锁竞争性能瓶颈

在多线程程序中锁竞争是最主要的性能瓶颈之一。在前面我们也提到过，通过使用锁来保护共享变量能防止数据竞跑，保证同一时刻只能有一个线程访问该临界区。但是我们也注意到，正是因为锁造成的对临界区的串行执行导致了并行程序的性能瓶颈。

阿姆达尔法则（Amdahl’s Law）

在介绍锁竞争引起的性能瓶颈之前，让我们先来了解一下阿姆达尔法则。我们知道，一个并行程序是由两部分组成的：串行执行的部分和可以并行执行的部分。假设串行部分的执行时间为S，可并行执行部分的执行时间为P，则整个并行程序使用单线程（单核）串行执行的时间为S+P。阿姆达尔法则规定，可并行执行部分的执行时间与线程数目成反比：即如果有N个线程（N核CPU）并行执行这个可并行的部分，则该部分的执行时间为P/N。由此我们可以得到并行程序总体执行时间的公式：

总体执行时间T = S + P/N。

根据这个公式，我们可以得到一些非常有意思的结论。例如，如果一个程序全部代码都可以被并行执行，那么它的加速比会非常好，即随着线程数（CPU核数）的增多该程序的加速比会线性递增。换句话说，如果单线程执行该程序需要16秒钟，用16个线程执行该程序就只需要1秒钟。然而，如果这个程序只有80%的代码可以被并行执行，它的加速比却会急剧下降。根据阿姆达尔法则，如果用16个线程并行执行次程序可并行的部分，该程序的总体执行时间T = S + P/N = (16x0.2) + (16x0.8)/16 = 4秒，这比完全并行化的情况（只需1秒）足足慢了4倍！实际上，如果该程序只有50%的代码可以被并行执行，在使用16个线程时该程序的执行时间仍然需要8.5秒！

从阿姆达尔法则我们可以看到，并行程序的性能很大程度上被只能串行执行的部分给限制住了，而由锁竞争引起的串行执行正是造成串行性能瓶颈的主要原因之一。

锁竞争的常用解决办法

避免使用锁

为了提高程序的并行性，最好的办法自然是不使用锁。从设计角度上来讲，锁的使用无非是为了保护共享资源。如果我们可以避免使用共享资源的话那自然就避免了锁竞争造成的性能损失。幸运的是，在很多情况下我们都可以通过资源复制的方法让每个线程都拥有一份该资源的副本，从而避免资源的共享。如果有需要的话，我们也可以让每个线程先访问自己的资源副本，只在程序的后讲各个线程的资源副本合并成一个共享资源。例如，如果我们需要在多线程程序中使用计数器，那么我们可以让每个线程先维护一个自己的计数器，只在程序的最后将各个计数器两两归并（类比二叉树），从而最大限度地提高并行度，减少锁竞争。

使用读写锁

如果对共享资源的访问多数为读操作，少数为写操作，而且写操作的时间非常短，我们就可以考虑使用读写锁来减少锁竞争。读写

锁的基本原则是同一时刻多个读线程可以同时拥有读者锁并进行读操作；另一方面，同一时刻只有一个写进程可以拥有写者锁并进行写操作。读者锁和写者锁各自维护一份等待队列。当拥有写者锁的写进程释放写者锁时，所有正处于读者锁等待队列里的读线程全部被唤醒并被授予读者锁以进行读操作；当这些读线程完成读操作并释放读者锁时，写者锁中的第一个写进程被唤醒并被授予写者锁以进行写操作，如此反复。换句话说，多个读线程和一个写线程将交替拥有读写锁以完成相应操作。这里需要额外补充的一点是锁的公平调度问题。例如，如果在写者锁等待队列中有一个或多个写线程正在等待获得写者锁时，新加入的读线程会被放入读者锁的等待队列。这是因为，尽管这个新加入的读线程能与正在进行读操作的那些读线程并发读取共享资源，但是也不能赋予他们读权限，这样就防止写线程被新到来的读线程无休止的阻塞。需要注意的是，并不是所有的场合读写锁都具备更好的性能，大家应该根据Profling的测试结果来判断使用读写锁是否能真的提高性能，特别是要注意写操作虽然很少但很耗时的情况。

保护数据而不是操作

在实际程序中，有不少程序员在使用锁时图方便而把一些不必要的操作放在临界区中。例如，如果需要对一个共享数据结构进行删除和销毁操作，我们只需要把删除操作放在临界区中即可，资源销毁操作完全可以在临界区之外单独进行，以此增加并行度。

正是因为临界区的执行时间大大影响了并行程序的整体性能，我们必须尽量少在临界区中做耗时的操作，例如函数调用、数据查询、I/O操作等。简而言之，我们需要保护的只是那些共享资源，而不是对这些共享资源的操作，尽可能的把对共享资源的操作放到临界区之外执行有助于减少锁竞争带来的性能损失。

尽量使用轻量级的原子操作

在例3中，我们使用了mutex锁来保护counter++操作。实际上，counter++操作完全可以使用更轻量级的原子操作来实现，根本不需要使用mutex锁这样相对较昂贵的机制来实现。

粗粒度锁与细粒度锁

为了减少串行部分的执行时间，我们可以通过把单个锁拆成多个锁的办法来较小临界区的执行时间，从而降低锁竞争的性能损耗，即把“粗粒度锁”转换成“细粒度锁”。但是，细粒度锁并不一定更好。这是因为粗粒度锁编程简单，不易出现死锁等Bug，而细粒度锁编程复杂，容易出错；而且锁的使用是有开销的（例如一个加锁操作一般需要100个CPU时钟周期），使用多个细粒度的锁无疑会增加加锁解锁操作的开销。在实际编程中，我们往往需要从编程复杂度、性能等多个方面来权衡自己的设计方案。事实上，在计算机系统设计领域，没有哪种设计是没有缺点的，只有仔细权衡不同方案的利弊才能得到最适合自己当前需求的解决办法。例如，Linux内核在初期使用了Big Kernel Lock（粗粒度锁）来实现并行化。从性能上来讲，使用一个大锁把所有操作都保护起来无疑带来了很大的性能损失，但是它却极大地 简化了并行整个内核的难度。当然，随着Linux内核的发展，Big Kernel Lock已经逐渐消失并被细粒度锁而取代，以取得更好的性能。

使用无锁算法、数据结构

首先要强调的是，笔者并不推荐大家自己去实现无锁算法。为什么别去造无锁算法的轮子呢？因为高性能无锁算法的正确实现实在是太难了。有多难呢？Doug Lea提到java.util.concurrent库中一个Non Blocking算法的实现大概需要1个人年，总共约500行代码。事实上，我推荐大家直接去使用一些并行库中已经实现好了的无锁算法、无锁数据结构，以提高并行程序的性能。典型的无锁算法的库有java.util.concurrent，Intel TBB等，它们都提供了诸如Non-blocking concurrent queue之类的数据结构以供使用。

陈冠诚

IBM中国研究院研究员，瑞典查尔姆斯理工大学（Chalmers）硕士，主要研究方向为并行计算、计算机体系结构及分布式系统。个人博客为并行实验室（http://parallellabs.com）。

责任编辑：高松（gaosong@csdn.net）

寻找机遇 创造未来 CSDN 热门职位全新推荐

欢网科技

huan 欢网科技

欢网科技急聘：

- 高级嵌入式开发工程师
- 高级需求分析师
- Android TV 系统架构师
- 系统架构师
- 高级java开发工程师

更多招聘信息和详细职位要求可登录欢网科技网站查询！
更多招聘岗位查询： http://www.huan.tv

简历投递：zhaopin@huan.tv
地址：北京市朝阳区劲松三区甲302号华腾大厦25层

欧特克软件（中国）有限公司

你一展身手的舞台，
打造美好未来

Autodesk

你一展身手的舞台，打造美好未来加入我们的队伍，以一种全新的方式实现设计理念。无论是设计可持续性建筑还是开发混合动力汽车，Autodesk软件都可助你一臂之力。作为全球领先的二维与三维设计、工程和娱乐软件提供商，我们致力于帮助打造出更美好的未来。

让我们共同创造未来!

更多职位信息请浏览网站：www.autodesk.com/careers
或直接将简历发送至：acrd.hiring@autodesk.com

搜狐畅游

搜狐 SOHU.com　畅游 ChangYou.com

游戏研发部3D程序专员:

工作职责：
1. 3D框架下各种功能模块和算法的实现；
2. 研究、开发和优化3D引擎。

职位要求：
1. 诚信正直，具备高度的责任心；2. 计算机相关专业本科及以上学历；3. 两年以上的3D引擎开发经验；4. 精通C++，面向对象编程，Windows编程；5. 熟悉3D图形学原理，熟悉3D几何；6. 精通DirectX/OpenGL，精通DX9；7. 具备良好的算法和数学基础，优秀编码习惯和学习能力。

公司网址：http://www.changyou.com
E-mail：gamejob@job.cyou-inc.com

北京壹人壹本信息科技有限公司

Android终端开发工程师
要求：
1. 大学本科及以上学历，计算机相关专业。
2. 熟悉Android系统架构和应用开发体系，有OS层、BSP层相关工作经验者优先。

JAVA/PHP平台开发工程师
要求：
1. 本科及以上学历，计算机相关专业。
2. 五年以上WEB开发经验，熟悉JAVA、PHP开发语言并精通其中一种。

联系人：曹卫　邮箱：caowei@eben.cn　电话：010—58204117
地址：北京市朝阳区建国路93号万达广场10号楼2506室。

方正国际软件有限公司

方正国际医疗卫生业务是中国首家医疗信息系统解决方案的提供商，经过16年的发展，已成为国内最大的数字化医院及区域医疗卫生网解决方案的提供商和服务商，并荣获“2011年度中国医疗领域最具影响力解决方案奖”，印证了方正国际医疗信息化领域的领军地位。

FOUNDER 方正

为适应业务的蓬勃发展，现在全国范围内诚征：

医疗信息化行业研发，工程项目，售前和销售精英

详细信息请登录智联，前程无忧或中华英才查询，并投递相应邮箱！公司主页：www.founderinternational.com

上海纵游网络技术有限公司

我们正在全力招聘中国人才，打造一个全新的中国团队，致力于中国智能手机游戏平台的构筑和运营。
愿意加入我们，一起构筑世界顶级的智能手机游戏平台吗？

DeNA China

Android/iPhone/Web application，**更多职位招聘信息请登陆公司主页进行查询。**

公司主页：www.denachina.com

请以“职位+姓名+CSDN”作为应聘邮件标题，将中英文或中日文简历发至hr@denachina.com

完美世界（北京）网络技术有限公司

完美世界（北京）网络技术有限公司（原完美时空）是中国领先的网络游戏开发商和运营商之一。公司成立于2004年，一直致力于创造优质的互动娱乐产业品牌，倾力打造拥有自主知识产权的高质量网游精品。

现诚聘

- ERP EBS 技术顾问
- flash(as3)开发工程师
- Java开发工程师
- 高级客户端开发工程师
- 游戏运维工程师
- PHP开发工程师
- 游戏服务器端软件工程师
- 游戏客户端软件工程师

职位详情请点击：http://special.zhaopin.com/bj/2011/wanmei03162823/index.htm
公司网址：www.wanmei.com　**电子邮箱：**zhaopin@wanmei.com

天津港保税区科技发展局

东软（天津）、展讯（天津）高薪诚聘IT英才

招聘职位：高级JAVA工程师，高级C++工程师，DSP工程师，芯片驱动工程师，应用软件工程师，驱动软件工程师，算法工程师

工作地点：

天津空港经济区（www.tjftz.gov.cn）

简历投递邮箱：resume@adm.tjftz.gov.cn

北京法国电信研发中心

Software engineer in the web service system development

Position requirements:

- Master degree in telecommunication, Compute Science OR equivalent
- Experience in J2EE, SOA, Spring, Hibernate, Struts, XML, REST, HTML5 development
- Experience in cloud computing development, such as SaaS, PaaS, etc.
- Strong programming skills, at least 3+ years experience in Java software development
- Familiar with the software development methodology
- Familiar with networking technology
- Good English communication skill
- Good team working spirit and communication skills, and hard working as well

Contact：hr@orange-ftgroup.com.cn

航天信息

JAVA/C++等研发岗位热招中

C/C++开发工程师

工作地点：北京

1. 计算机软件或相关专业统招本科以上学历
2. 具有2年以上开发经验，熟练掌握.NET、C++ Builder/Delphi开发工具，可熟练进行客户端和服务器端软件开发
3. 理解COM组件的架构原理，具备一定的COM编程经验，能够进行COM\ActiveX\DLL开发
4. 熟悉Web Service等网络通讯开发
5. 有良好的理解、沟通能力，良好的文档能力

请以职位-姓名-CSDN形式命名邮件，发至：hr@aisino.com

爱立信（中国）通信有限公司

爱立信专注于通信研发130余年，始终屹立无线通信领域行业领先的地位。而其多媒体软件系统更是独树一帜，成为爱立信的主要战略方向。爱立信公司多媒体事业部上海的研发中心正肩负这样的使命，其中的TV（包括IPTV，WebTV，CableTV和MobileTV）产品线更是战略重点，为此我们诚邀您加盟。

我们这次主要招聘 3 类职位：

1.MobileTV/WebTV的软件开发职位

2.TV解决方案的集成和测试

3.TV机顶盒的集成和测试

广州瀚信通信科技股份有限公司

招聘职位：

- C/C++ 系统架构师
- C/C++ 网络开发工程师
- Silverlight/WPF 前端交互设计师
- C# 开发工程师
- 软件测试工程师

详细信息请登录CSDN网站查询！

请以“职位+姓名+CSDN”作为邮件标题，
将简历发送至：yfhr@hantele.com　**公司主页：**www.hantele.com

借助模糊测试
深耕细作你的压力测试

文 / 张小明

本文通过将模糊测试应用到具体邮件服务器安全产品的压力测试之中，介绍了如何设计和进行模糊测试，并把模糊测试和全球化随机测试相结合以求得更广的测试面积，最终达到更好的压力测试效果。

模糊测试（Fuzz Testing）一直被黑客们广泛使用，却很少应用于商业软件系统开发过程中。在互联网安全产业，如何让模糊测试重新焕发活力，是我一直在思考的问题。

概念及目的

在计算机领域，模糊测试是一个古老却容易被忽视的测试领域，它的诞生绝对早于真正意义上的自动化测试。甚至可以这么说：自从计算机允许用户输入，模糊测试便应运而生了。

所谓模糊测试，顾名思义可以理解成用一段非常规的字符串（或值）去替换正常的输入值，或者用非常规的、不合法的文件替换合法的文件输入，然后检测系统的行为是否正常、模块是否被破坏。虽然概念简单，但模糊测试的作用是非常巨大的，有时它能帮助发现严重的安全漏洞，这也是它深受广大黑客喜爱和认可的原因。

模糊测试在黑客界比较流行，凡是有输入点的地方总有黑客存在。基于对被攻击系统的学习和研究，黑客们会尝试有目的地设计出各种不同的随机输入值去攻击系统直至找到系统的漏洞。SQL Injection Attack便是黑客们常用的一个模糊测试的经典案例。

为了消除被攻击的风险，必须先发制人，在产品发布之前更多地发现产品的安全问题和漏洞便成了软件测试工作者关注的问题。

应用场景

我们的一款商业邮件服务器安全防护系统，目前用于测试的基本邮件库里大约有20万封测试邮件样本，扩展邮件库里大约有100万封。作为普通的压力测试之用，这些样本看起来已经足够多，而且邮件样本在不断更新，但从模糊测试的角度看，依然有以下不足点。

邮件库里的邮件样本是固定的，系统的输入点就是这些邮件的发送和接收，然后对邮件进行分析，固定的量不能覆盖更全、更完整的测试面。

在版本跟进过程中总是使用固定的样本不符合日新月异的测试需求，容易产生测试惰性。所谓惰性，就是固定的样本总是发现相同的问题，而这些问题早在之前就被发现了，再用同样的样本做压力测试很难发现新问题和新漏洞。

以上两个不足点的共同之处就在于样本是固定的。对此，我们的解决方案就是在压力测试中引入模糊测试，基于固有的测试样本基数模糊生成更多甚至成指数倍递增的模糊样本库，而且模糊的策略不同，模糊出来的样本也不尽相同，这便实现了我们测试的初衷——让样本更多、更动态。

在压力测试中如何进行模糊测试

压力测试中引入模糊模块在流程上是一个很简单的概念，就是将原始的输入样本进行

一番模糊，生成更多的样本，在本文应用场景下，便是基于已有的邮件样本库然后模糊出来更大的样本库，示意流程如图1所示。

其中①便是压力测试系统新增的Fuzz模块，通过Fuzz模块扩容后的邮件样本库（扩容倍数会依据Fuzz策略不同而定）在压力测试用例分发程序分发样本到指定的已安装了邮件安全产品的邮件服务器上去。

其中②表示分发程序根据样本分发策略具体进行分发的过程。

其中③表示新引入的自动化测试流程，基于Fuzz的压力测试后，产品需要进行安全漏洞和功能性验证，此处需要启动的自动化测试范围需要根据压力测试的策略和目标而定，最常见的比如：

- 没有程序和服务的崩溃与挂起。
- 没有内存泄漏。
- 通过最基本的（优先级别最高的）测试用例。

一般说来，模糊测试和压力测试中发现的问题都具备以下特点：

- 严重的安全漏洞和程序异常。
- 普通人工测试和自动化测试很难复现的bug。
- 比较难解的bug。

④在普通的功能性自动化测试验证压力测试结构之后，再有目的地做一些探索性的测试。

图2中给出一个我使用WPF设计的UI原型图，配合上述流程图，可以看出四大块功能：

Fuzz选项，被Fuzz的样本库配置，目标测试服务器配置，发送（启动测试）选项。

Fuzz功能设计

不同的项目和产品，不同的测试目标，Fuzz的策略和算法也不尽相同，但理念是一致的，那就是把正常的输入混淆以生成更多的输入值。比较通用的做法就是字节替换。字节替换意味通过将文件或者字符串中的字节替换成策略中的预设字节以达到Fuzz的效果， 如图3中给出了Fuzz选项示意。

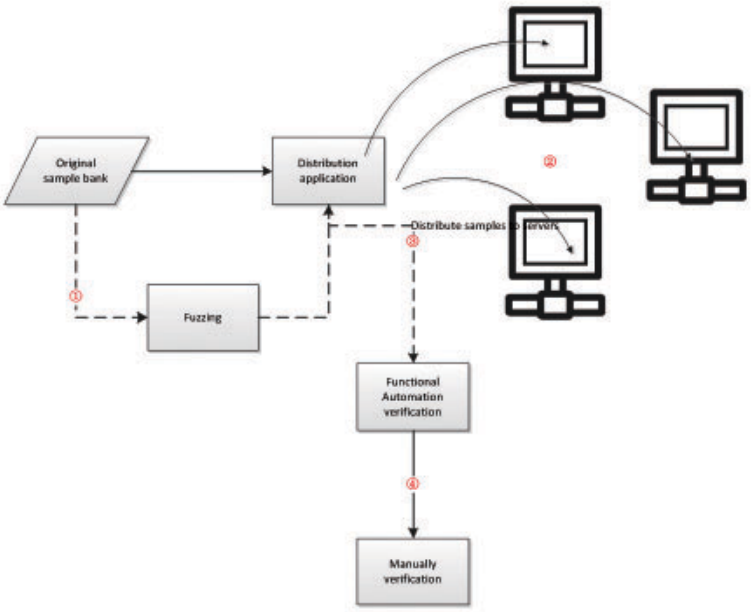


图1　模糊测试流程图

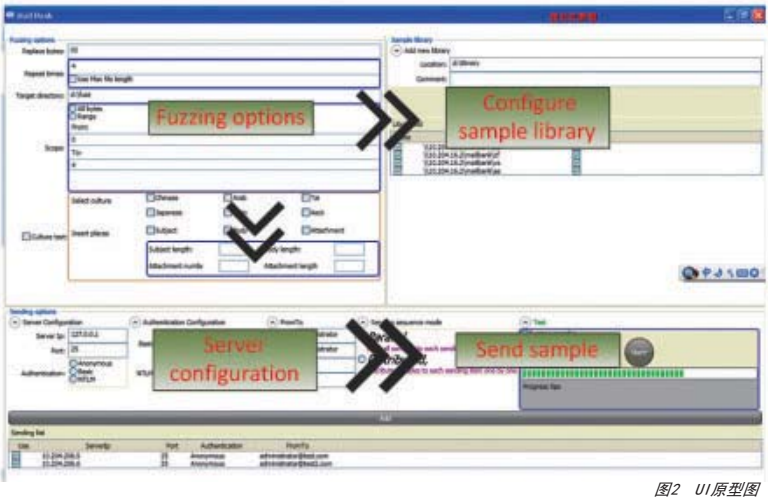


图2　UI原型图

其中：

Replace bytes：要替换为的字节。

Replace times：替换的次数，意味着样本里有多少字节会被替换成目标字节，如果选中Use Max file length， 意味着文件中所有字节都会被替换一边，假设文件长度是10个字节，那一份样本 Fuzz之后就是10份样本。

Target directory：Fuzz之后的样本保存路径。

Scope：Fuzz范围，因为有些文件太大，如果是针对所有字节全Fuzz 一边会造成测试数据过于庞大，所以一般针对大文件会指定Fuzz的字节范围。

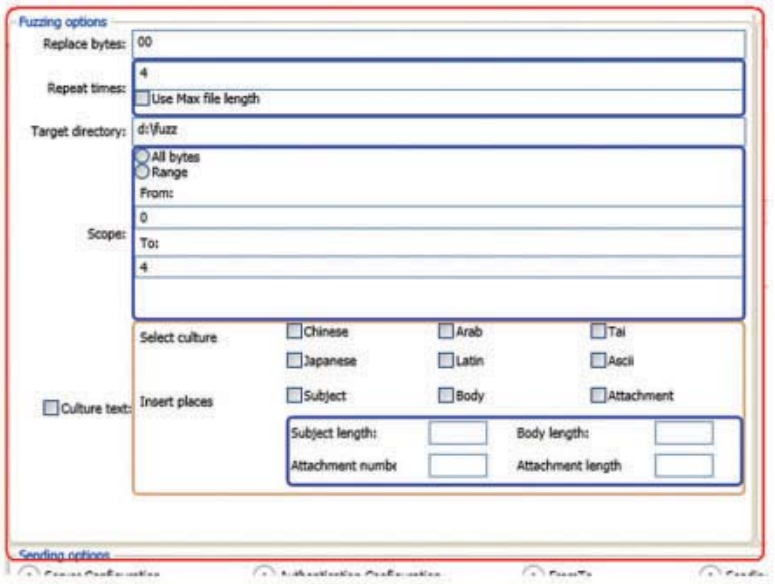


图3 Fuzz选项示意图

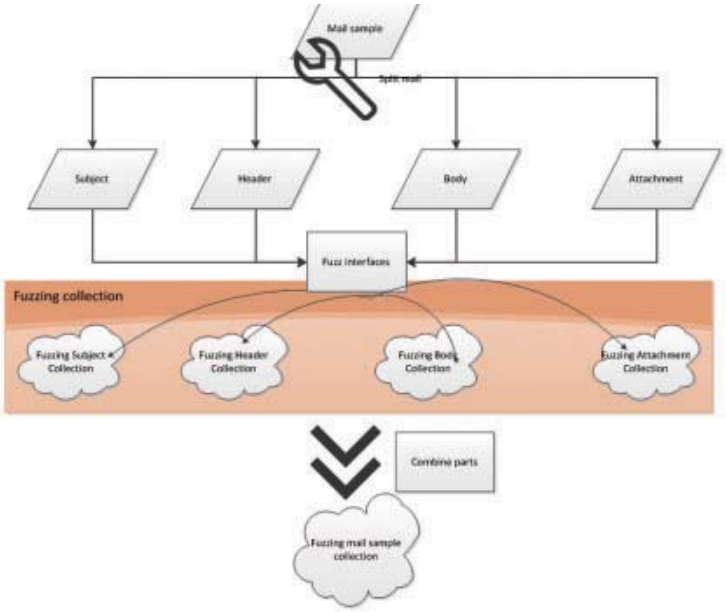


图4 邮件样本Fuzz示意流程图

Culture text：针对全球化测试的需求，需要测试多语言文化下的系统功能，所以在Fuzz的同时并随机生成指定的语言文化字符串以应测试之需，更广地覆盖了测试范围。该部分会在后面部分再次探讨。

本文所用到的邮件样本库里的样本是以EML格式存在的 （EML格式是微软公司在Outlook中所使用的一种遵循RFC822及其后续扩展的文件格式， 并成为各类电子邮件软件的通用格式)， 在Fuzz之前要进行EML文件解析， 一封邮件包含subject、 header、 body、 attachment四个部分， 于是Fuzz的任务便拆分成了分别Fuzz邮件的四个部分，每一个部分分别Fuzz之后再与其他部分的结果排列组合成新的更多的样本。示意流程如图4所示。

下面是一段load Eml格式邮件的代码，实例化一封邮件，获得邮件各个组成部分的对象属性。

```
            public static CDO.Message
LoadMessageFromEml(string url)
        {
                ADODB.Stream adoStream =
null;
                adoStream = new ADODB.
Stream();
            adoStream.Open(Type.Missing,
ADODB.ConnectModeEnum.adModeUnknown,
                                ADODB.
StreamOpenOptionsEnum.
adOpenStreamUnspecified, string.Empty,
string.Empty);
            adoStream.LoadFromFile(url);
               CDO.Message message = new
CDO.Message();
                  message.DataSource.
OpenObject(adoStream, "_stream");
            adoStream.Close();
            return message;
        }
```

强化Fuzz testing力度，引入Culture random test

商业产品尤其是针对国际市场的产品需要全球化本地化测试，这涉及多语言文化的数据测试，由于我们无法覆盖某语言文化下的全部字符，故使用随机生成测试数据的方式尽量弥补这一缺陷，这样可提高测试的覆盖面积。在模糊测试的环境里，随机的测试数据也属于Fuzz概念的一部分，所以除了上述普通的Fuzz算法策略，我们考虑结合多语言文化的随机测试，即：在Fuzz的基础上，给测试数据填充多语言文化的随机字符串，在Unicode编码规范里，UTF-8作为它的表现形式容纳了全世界所有的语言文化字符（当然，不是绝对的），而每一种不同的语言字符集被安置在不同的固定区域内，在固定的区域内随机出来的位置就是针对该语言文化下一个随机的字符，Unicode相关知识可参见http://www.unicode.org/，这里不做赘述。

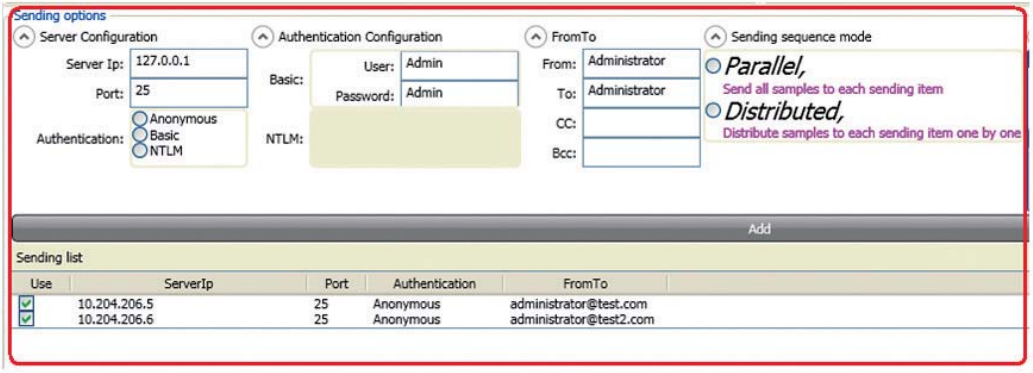


图5　发送选项的配置部分

细化Fuzz testing粒度，区分对待不同文件类型

邮件附件的Fuzz存在文件格式的问题，如果都按照文本文件Fuzz策略进行Fuzz，势必会导致很多格式化文件Fuzz达不到应有的效果，反而破坏了正确的文件格式，所以更细化的Fuzz应该因文件类型而异。当然，如果测试者的目标就是测试文件格式是否被兼容则另当别论。在本实战中，测试目的是Fuzz出来更多可用的格式正确的邮件。

常见的邮件格式化文件类型的附件有MS Office文件和Adobe文件，首先把常见的格式整理出来，然后通过相应的格式解释器去解析格式化文件中的具体内容，比如通过微软Office文件便提供Office编程模型解析相应的Office文件，然后针对解析出来的内容进行Fuzz，做到测试目标明确，节省测试精力。

发送选项及结果验证

Fuzz之后便是启动压力测试以及测试的验证部分，在文中的测试环境中，需要在不同的邮件服务器上分发测试数据，如图5所示是发送选项的配置部分。

其中，分发应用程序设置两种发送模式：

Parallel， 并行模式，同时把所有的样本发送到所有可选的服务器上。

Distributed， 分布式模式，为节约时间（经过Fuzz之后的样本会相对原来的量有指数倍的增长），把样本平均分配发送到不同的服务器上。

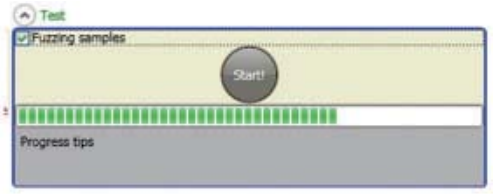


图6　启动全程测试界面

一个start！按钮带你启动全程测试：

然后就等着收获bug果实吧！

关于如何验证的流程，这里也做了创新，一般压力测试之后的验证会通过人工去检查程序或服务是否异常或是否产生内存泄漏，事实上这些都可以（应该）通过自动化测试去实现。而且，应该跑一轮功能性的自动化测试去验证在压力测试之后系统的基本功能是否还正常工作。最后再有目的地做探索性的测试。

张小明

英文名Sayid， 趋势科技南京研发中心软件测试工程师，专注于自动化测试相关的研究。

责任编辑：高松（gaosong@csdn.net）

JavaScript的回调机制讲解

文 / 李叶青

本文尝试用几个例子说明异步通信的环境用JavaScript写回调函数与C语言的区别，以及为什么JavaScript原生更适合做这件事情。

由于其运行环境的特殊性，JavaScript大量使用异步的通信机制，凡是涉及网络调用和事件机制的代码都会涉及。在异步通信的环境下编码经常会用到回调函数。JavaScript由于有函数式语言的一些特点，实现回调函数非常优雅和自然，包括函数作为一级的对象、匿名函数、闭包机制等。但是要体会到个中的优雅，需要先融汇贯通这些机制。如果是初学者学习这些东西可能比有编程经验的人少很多障碍，认为事情本来就该是这个样子。但是，对于长期使用过程式语言编码（比如传统的C/C++程序员），又没有接触过函数式语言的程序员来说，可能需要越过一道思维的小坎儿。这件事情有时候会造成一定的困扰，因为“老手”程序员会想：毕竟我已经懂得一套能写程序的方法，大家都说语言之间差别不重要，毕竟C++里面也有使用异步调用的时候，主要注意一下语法的区别就好了。所以最终就变成了使用JavaScript来模仿别的过程式语言，这样的结果最终很有可能是写出很别扭的程序给自己添堵。本文尝试用几个例子说明异步通信的环境用JavaScript写回调函数很使用类似C语言写回调函数的区别，以及为什么JavaScript原生要更适合做这件事情。（简单起见，下面例子中的代码均为伪代码，并不一定严格符合C/C++或者JavaScript的语法，但是笔者尽量写得与语法要求接近。）

我们首先从C/C++的同步调用开始，假设我们要写一个函数，向远方的服务器发送一个字符串形式得命令，并且从服务器得到一个字符串作为响应。例1就展示了使用C语言在同步通信的机制下代码的样子。

例1 使用C语言的编码方式实现调用访问远程的接口：

```
//{{{get_data_v1
int get_data_v1()
{
    // 准备数据
    char bufCmd[]="cmd=1001&uin=123456&p
aram=abc";
    char bufRcv[4096];
    // 建立连接
    socket s = new Socket();
    connnect(s, ip, port);
    // 发送数据
    send(s, bufCmd);
    // 接收数据
    recv(s, bufRcv);
    // 处理结果
    use(bufRcv);
    return 0;
}
//}}}
```

在例1中，get_data_v1执行了准备数据、创建了socket、建立连接、发送请求、接收响应并最终使用use函数处理接收到的数据，一切都显得很自然。为了方便说明问题，我们将这个通信的过程封装一下，将整个建立连接并收发包的过程封装成一个叫send_and_recv的函数。

例2 将通信过程封装成独立的函数，简化业务流程代码：

```
//{{{get_data_v2
// 发包收包的过程
int send_and_recv(struct addr, char*
bufCmd, char* bufRcv)
{
    socket s = new Socket();
    connnect(s, addr.ip, addr.port);
    send(s, bufCmd);
    recv(s, bufRcv);
}
// 原来的业务流程
int get_data_v2()
{
    // 准备数据
    char bufCmd[]="cmd=1001&uin=123456&p
aram=abc";
    char bufRecv[4096];
```

```
    // 通信，收发数据
    // addr={ip, port}
    send_and_recv(addr, bufCmd, bufRcv);
    // 处理结果
    use(bufRcv);
    return 0;
}
//}}}
```

例2和例1很类似，不过是对通信过程进行封装了，并且ip-port对也变成了一个叫addr的地址结构体。改动以后处理过程变得更简单，剩下准备数据、通信和处理结果三步。现在，我们开始进入正题，现在我们假设这个通信过程变成异步的，它接收一个回调函数用于处理取得的数据。如例3所示。

例3 将通信过程变成异步调用：

```
//{{{get_data_v3
// 变成异步调用以后，原来的调用过程分成了两段
// 前半段组装参数调用发包过程
// 后半段处理返
// 这里假设send_and_recv是一个异步的网络通信函数
void get_data_v3()
{
    char bufCmd[]="cmd=1001&uin=123456&param=abc";
    char bufRcv[4096];
     send_and_recv_async(addr, bufCmd, bufRcv, callback);
} // end of get_data_v3

// 回调函数的定义
int callback(char* bufRcv) {
    // 处理接收都的数据
    use(bufRcv);
    return 0;
}
//}}}
```

在例3中，假设使用了一个异步的通信过程send_and_recv_async，最后一个参数callback是一个回调函数指针。然后，当接收到响应以后，send_and_recv_async会调用callback并传入接收到的数据。相比例2，这个get_data的过程被异步通信过程一分为二：前半段为准备请求，后半段是处理结果。事实上，对将同步通信方式变成异步以后，都会涉及到将原来完整处理过程一分为二的问题。在两段程序没有什么相互依赖的情况下，这样的分解不会造成什么问题。但是，如果处理结果的过程依赖于一些外部参数，那么情况就会变得很复杂。我们先来看看在同步通信的情况下，程序的样子，见例4。

例4 假设处理结果的时候依赖外部参数：

```
//{{{get_data_v4
//  这里原来的业务流程需要外部传进来的两个参数（a, b）来决定如何处理结果
int get_data_v4(int a, int b)
{
    char bufCmd[]="cmd=1001&uin=123456&param=abc";
    char bufRcv[4096];
    send_and_recv(addr, bufCmd, bufRcv);
    // 处理过程依赖于外部传进来的参数a和b
    use(bufRcv, a, b);
    return 0;
}
//}}}
```

在例4中，我们的结果处理过程use依赖于传入的两个参数a和b。现在我们来看看例4的程序如果使用异步通信会怎样，见例5。

例5 加上参数依赖后再变成异步调用：

```
// 版本a
//{{{get_data_v5
// 需要参数的异步调用需要将参数透传到后半段的回调函数中
void get_data_v5a(int a, int b)
{
    char bufCmd[]="cmd=1001&uin=123456&param=abc";
    char bufRcv[4096];
     send_and_recv_async(addr, bufCmd, bufRcv, callbacka, a, b);
} // end of get_data_v5a

// 回调函数的定义
int callbacka(char* bufRcv, int a, int b)
{
    use(bufRcv, a, b);
    return 0;
}

// 版本b
int g_a;
int g_b;
void get_data_v5b(int a, int b)
{
    g_a = a;
    g_b = b;
    char bufCmd[]="cmd=1001&uin=123456&param=abc";
    char bufRcv[4096];
     send_and_recv_async(addr, bufCmd, bufRcv, callbackb);
} // end of get_data_v5b

// 回调函数的定义
int callbacka(char* bufRcv, int a, int b)
{
int callbackb(char* bufRcv) {
    use(bufRcv, g_a, g_b);
    return 0;
}
//}}}
```

例5中有两个版本，get_data_v5a假设了通信机制可以透传a和b两个参数给回调函数，get_data_v5b则使用了两个全局变量来传递处理结果所需的参数。两个都不见得是很好的方法，get_data_v5a的问题是，异步通信的机制不

见得能提供这种透传机制，除非程序员自己封装；即使程序员自己封装，那也意味着如果要实现多个处理数据的过程（像get_data）那就要实现多个异步调用的过程（send_and_recv_async），代码复杂且复用性差不好维护。而全局变量的版本也好不到哪里去，使用这种全局的机制，意味着不必要的信息暴露，也就有被别的地方错修改的问题，同时这个函数还变成不可重入的。即使将全局机制封装在一个类里面，每次初始化一个对象，可以改善依然不能解决信息暴露的问题，同时还带来了管理这多个对象的复杂性。

两种方法相比而言，貌似透传的机制要稍好一些。我们对get_data_v5a略做修改，使得它通信过程能够有更广泛的复用。

例6 使用closure对象打包过程中的参数：

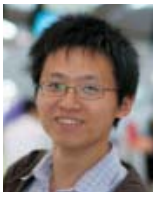
```
//{{{get_data_v6
// 为了统一回调函数的形式并且缩短回调的参数列表，将这种需要透传的参数只有一个
// 统一的数据结构打包
void get_data_v6(int a, int b)
{
    // 准备数据
    char bufCmd[]="cmd=1001&uin=123456&param=abc";
    char bufRcv[4096];
    // 打包处理结果所需要的参数
    closure.a = a;
    closure.b = b;
    // 通信
    send_and_recv_async(addr, bufCmd, bufRcv, callback, closure);
} // end of get_data_v6

// 回调函数的定义
int callback(char* bufRcv, struct closure) {
    // 处理结果
    use(bufRcv, closure.a, closure.b);
    return 0;
}
//}}}
```

例6里面使用了一个叫closure的结构，假设这个结构是个通用的数据容器，可以容纳我们使用的个中类型的任意数量的参数。增加了这一个万能的数据容器参数以后，异步通信过程只要能透传这么一个数据容器就能够很好支持个中各样的参数透传的需求。这个数据容器由于是在get_data函数内部产生的局部变量，不会污染全局数据或者比get_data更大的作用域。这种受限的可见性不仅提高了代码的可维护性，还恢复了函数的可重入性。

至此我们关于回调机制的实现的假想代码可以说已经达到比较优雅的程度了，仅仅还有一朵小乌云。那就是我们忽略了C/C++语言里面并没有原生实现这个超级结构，同样我们依然还有一点点麻烦就是还需要指定要透传的参数。考虑到原本从准备数据到通信再到处理结果是一个完整统一的过程，原本不需要区分什么数据是前半端使用的什么数据是后半段使用的，只要让前半端和后半段共享一个上下文在大部分情况下就能满足需求了。所以现实情况下我们只能做一些妥协，使用个中折衷方案来使得程序能运行起来。同样，考虑到回调函数和启动函数的关系，给回调函数命名也不是那么优雅的事情，因为毕竟它们只是同一个过程的两半，却要使用两个名字，合理一点就应该叫get_data_first和get_data_second，或者get_data_trigger和get_data_result_handler。如果接口多的话，就会有很多这种某过程first和某过程second，或者某过程trigger和某过程result_handler。能不能某过程就象同步那样使用一个名字呢？我们的设想真的就没有办法达到吗？答案是否定的，在Javascript能够帮助我们实现我们所有的设想。

对编码方式的影响

使用JavaScript来使用异步通信机制或者其他异步机制时，应该像同步代码一样编写，将准备数据、通信以及处理结果三步放到一个完成的流程下面，保持代码逻辑的高内聚。代码也不至于因为使用了异步通信的机制而变得四分五裂。从上面例子可以看到，这个过程需要做的仅仅时将数据处理的部分封装在一个匿名函数中，程序员不需要关心数据透传的过程，更加不应该越俎代庖地去实现任何过程数据地保持以及数据透传的工作。因为多一行代码就多一个可能的错误来源。



李叶青

腾讯公司Web前台开发工程师，任职于互联网系统部门。关注Web前端技术、架构和设计以及移动开发。

责任编辑：高松（gaosong@csdn.net）

新书上架

本月新书

Android开发秘籍

作者：James Steele/Nelson To

将功能丰富、结构复杂的Android应用程序，通过实例详尽解释，并提供大量可用代码，极大提高了实际项目的构建效率。

软件研发之道

作者：Jim McCarthy/Michele McCarthy

书中主角是微软Visual C++开发团队，介绍了他们如何组建优秀的软件开发团队，如何在一定时间内成功交付软件。

Git权威指南

作者：蒋鑫

可谓关于Git的百科全书，作者对Subversion和Git等版本控制工具有十分深入的研究，参与了Git以及Gitosis、Topgit、Gistore等与Git相关的开源软件的开发或创建。

Java编程思想（评注版）

作者：Eckel，B.

学习Java的路不好走，因为还有无数的技术需要学习，此评注版本，非常值得反复阅读体会。能够帮助读者跨越Java的重重险阻，领略高处才有的壮美风光。

创业在微软：微软亚洲工程院成长启示

编著：许凤婷

本书详尽勾勒了微软亚洲工程院的成长历程，生动展现了工程院的独特创业文化，以及张宏江博士富有魅力的领导艺术。

推荐图书

用户体验要素：以用户为中心的产品设计（原书第2版）

作者：Jesse James Garrett

译者：范晓燕

出版社：机械工业出版社

书名曰“用户体验要素”，但除了开篇讲述一位用户在经历一连串相当糟糕的产品后的狼狈，还有最后的章节，概括地描述了整本书与用户体验的结合。其他讲的都是网站构建的步骤和要素，读者可以基本上了解建立一个网站所需要经过的步骤。

然而，并非作者偏题，而是与设计理念相关。“用户体验”并非指一件产品本身如何工作（因而描述性的讲解无异于隔靴搔痒），而是指产品如何与外界发生联系并发挥作用，这是与产品的构建一同发生的。在创建良好用户体验的过程中，考量的就是以用户为中心。在此，开发产品的每一步骤，都要把用户列入考虑范围，然而仅靠灵感是无法做到面面俱到的。

作者用大篇幅讲述的就是把用户体验变成一件可以系统控制和评估的工作，可操作性大大提高，对照用户体验的要素列表，可以一项项检查关键的设计。作者将用户体验中一个个大问题分解成小问题，再提出针对性解决方案，避免了缺乏整体考虑导致的局部不协调。在此基础上，作者提出了用户体验的五个层面。

表现层：用户所能看见的一切，字体的大小，导航的颜色，整体给人的感觉。

框架层：决定某个板块或按钮等交互元素应该放在页面的什么地方。

结构层：用来设计用户如何到达某个页面，并且在他们做完事情之后能去什么地方。

范围层：结构层确定网站各种特性和功能的最佳组合方式。

战略层：网站的范围基本上是由网站的战略层所决定的。

以上五个层次的提出，保证了在设计用户体验的过程中，不会遗漏至关重要的事情，需要处理的事务也不至于因为太过抽象而无从下手。并且，每一层都由位于其下的决定，这种耦合性让一些具体而微的决定将变得顺理成章，减少了大量设计精力，当然，这对更底层决策要求更加慎重。如今互联网产品关联了生活的方方面面，重要性不言而喻，而本书所涉理论知识，为很多互联网产品经理提供了创造良好用户体验的有力武器。

重构：改善既有代码的设计（评注版）

编著：Martin Fowler/Kent Beck/John Brant/William Opdyke/Don Roberts

评注：张逸

出版社：电子工业出版社

近十年来，若要讨论如何改进代码的质量，很难绕过Martin Fowler的这本经典著作。这本书已经影响了几代程序员。但遗憾的是，在现实中我们仍然看到了重构的步履维艰。

重构，一言以蔽之，就是在不改变外部行为的前提下，有条不紊地改善代码。多年前，正是本书原版的出版，使重构终于从编程高手们的小圈子走出，成为众多普通程序员日常开发工作中不可或缺的一部分。本书也因此成为与《设计模式》齐名的经典著作。

今天，无论是重构本身以及业界对重构的理解，还是开发工具对重构的支持力度，都与本书最初出版时不可同日而语，但书中所蕴含的意味和精华，依然值得反复咀嚼。

本评注版力邀国内资深专家执笔，在英文原著基础上增加中文点评与注释，大多数点评内容，并非只言片语，很多都是点评者重构心得以及重构技巧的运用。点评者阅读参考了大量的书籍，例如《程序员修炼之道》、《重构与模式》、《领域驱动设计》、《反模式》、《软件架构的艺术》、《修改代码的艺术》、《代码整洁之道》等十余部相关书籍。毕竟站在软件世界的角度来看，Martin Fowler写作本书的时代已经相当“古老”了。在这之后，产生了许多精彩的设计技巧、重构理念与方法。譬如在与重构相关的内容中，本书未曾论述的就包括：架构重构、界面重构、数据库重构、重构模式等内容，而这些都会在点评中得以体现。

Cassandra权威指南

作　者：Eben Hewitt

译　者：王　旭

出版社：人民邮电出版社

以Facebook和Twitter为代表的Web2.0应用的数据爆炸性增长为后台存储系统带来了两个新挑战：海量数据存储和数据查询低延时。为了设计实现这样的数据库架构，Google和Amazon分别给出了自己的实现技术。Google开发了BigTable，Amazon开发了Dynamo。这两种实现都具有高扩展性和高可用性，并且可以在廉价的服务器中稳定运行，节约了大量硬件成本。Facebook基于Google和Amazon的研究成果开发出了Cassandra，不仅具备BigTable强大的数据模型，还兼具Dynamo的简洁性，例如端到端数据备份，以及强大的错误容灾功能等。

2008年，Facebook将Cassandra贡献给开源社区。大量公司和机构相继开始使用它作为建构实时应用程序以处理海量数据的首选，其中包括Netflix、Twitter和Cisco等公司。

本书作者曾任阿里巴巴数据仓库开发工程师，参与设计和开发了多个基于Cassandra的大型应用。书中他系统讲解了Cassandra的配置、应用、编译等方法，还从源代码的角度对其实现原理和底层的工作机制作了分析与探讨。

Cassandra的开发社区非常活跃，它在0.6版中与Hadoop进行了整合，使得Cassandra中的数据能够被Hadoop处理。事实上，尽管Cassandra可以处理海量数据的实时读取与写入，并且在二级索引的支持下，提供对值的查询功能，但它仍缺乏对数据进行分析的能力。而Hadoop虽然提供了海量存储功能，但其一次写入、多次读取，不支持数据修改的特性，仍是一些应用的瓶颈。

而本书最为重点的部分即是讲解了如何将Cassandra的存储能力与Hadoop MapReduce的计算能力整合，在两者间完成数据交换，进而得到相应的计算结果。

Datastax公司作为Cassandra的主要开发成员，开始为使用者提供服务支持，并开发出一系列工具，进一步提高了Cassandra的实用性。2011年5月，Datatax提供了一款振奋人心的工具：Brisk。通过它，Cassandra将取代Hadoop项目中HDFS和HBase，直接与MapReduce和Hive项目集成，提高整个海量数据分析系统的性能和易用性，Canssandra的发展前景不可限量。

全球排行榜

Amazon

01 Visualize This
02 In The Plex
03 Pro ASP.NET MVC 3 Framework
04 The Information: A History, a Theory, a Flood
05 Head First Java, 2nd Edition
06 Metasploit: The Penetration Tester's Guide
07 Learning Cocos2D
08 Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition
09 Programming in Objective-C (3rd Edition)
10 Head First HTML with CSS & XHTML

天珑书局

01 前进Android Market！Google Android SDK实战演练
02 深入浅出Android系统移植与开发测试
03 Google Android SDK开发范例大全
04 探索iPhone 4程序开发实战
05 全球最强VMware vSphere 4企业环境建构
06 培养与锻炼程序设计的逻辑：世界级程序设计大赛的知识、心得与解题分享
07 HTML5：建置与执行
08 王者归来：用Linux移植各种硬件
09 App程序设计入门：iPhone & iPad
10 马上就能用！Android SDK程序代码即可贴

第二书店

01 深入理解Java虚拟机：JVM高级特性与最佳实践
02 Hadoop权威指南（第2版）
03 0day安全:软件漏洞分析技术
04 大话数据结构
05 云计算安全与隐私
06 谁说菜鸟不会数据分析
07 Android技术内幕：系统卷
08 疯狂Android讲义
09 Git权威指南
10 Java编程思想

GEEK产品

*部分产品由麦极网（www.mygeek.cn）提供

Lethal Pro Gadget Spider Stand

手持设备一般都非常方便，但是当有的时候需要用手干别的事情的时候就麻烦了，譬如做饭、开车等。这个支架采用铝合金和碳纤维制成，专门用来替你拿住东西。可以固定住电话、平板电脑、相机或者其他数字设备，然后用蜘蛛状的触脚挂到任何位置上去。

Rollei Bullet HD Action Camera

Rollei Bullet高清运动相机外型酷似子弹头，设计专门用于固定在头盔或自行车等户外设备上，用来拍摄运动视频的一款小型相机。该款相机具有170度的广角镜头，可以录制730P的高清视频，内置miniSD卡可扩充至32GB 的存储。

Verbatim Bluetooth Mobile Keyboard

Verbatim蓝牙键盘是专为平板电脑和智能手机设计的。布局类似笔记本键盘，中间可折叠，留有一个托架可放置手机或其他设备。

Homade“迷你电视机”

Homade出品的又一款迷你家电，不过这次设计师没有选择时尚的东西，而是把20世纪80年代那种方盒子电视机拿来作为灵感，停台之后的画面设计为闹钟的显示界面，相当有怀旧的色彩！除了不能看电视，它能耐真不小呢，几个按键都是巧妙结合了电视机的外形和闹钟的功能，有四种悠扬的音乐闹铃，让你在悦耳的音乐中醒来。此外，还有设置键，可以设定闹钟模式。

福特心率监测座椅

这是一款福特公司正在研发的心率监测座椅。利用靠背表面的六个特殊传感器，它能够测量驾车者的心脏脉冲信号，并针对潜在的心血管问题发出预警。实验表明，即便隔着10层棉布，这些传感器也依然能获得强劲的脉冲信号。来自德国亚琛工业大学的专家指出，由于驾车者会长时间地保持一个相对平静的姿势，体力消耗也比较少，这使得车内成为监测心脏活动情况的理想环境。

VIO light全球首款iPhone/小电器消毒器

此刻，有超过1亿个病菌潜伏在手机的键盘、听筒、美丽的壳上！这些潜伏的家伙，都是随时可能威胁身体健康的隐患。每天对手机消毒，绝对有必要。不过，单纯拿酒精擦擦，并不能完全消灭那些顽固的家伙们，这需要更强悍的武器——VIO light紫外线电子设备消毒器。它操作起来非常简单，只要从底座上取下银盖，将手机放到消毒室里，盖上盖子之后消毒自动开始。看到蓝色的光带脉动打开不到5分钟之后，蓝灯关闭表示消毒完毕。

Arduino Uno电子积木

Arduino 是一款开源“硬件”，也就是说这款产品的硬件设计包括参考实现的电路图都是开放的，硬件电路板可以自行焊接组装，也可以购买已经组装好的。功能方面，计算机可以通过USB连接到Arduino上，通过其IO接口实现对外围设备的交互控制。Arduino Uno是主控板，除此之外还包括不少外围的辅助板，包括各类传感器、网络模块、蓝牙模块、通信模块等，可以在此基础上自由组合配置。

Dream Cheeky USB Friends Alert邮件提醒器

谁还用老旧的邮箱？来，一起怀个旧吧。这个小东西不但能帮你提醒邮件状态，还居然能够支持Facebook、Twitter、Skype、MSN的消息呢。哦，差点忘记了，国内只有MSN你偶尔可以玩一下。

JavaScript和HTML一样将长存

John Resig访谈

整理 / 本刊编辑部

在参加完CSDN组织的TUP对话大师系列演讲活动后，27岁的jQuery之父John Resig接受了本刊总编刘江的深度访谈，这篇对话文章，让我们一窥这位著名程序员的人生及技术感悟。

John Resig认为，JavaScript和HTML一样会长久存在，20年内开发者肯定会一直用JavaScript写网页应用程序

编程初体验

《程序员》：你是如何开始编程的？

John Resig：第一次编程大概是在初中，14、15岁，当时有个朋友带来张软盘，里面有QBASIC。在DOS系统下他向我展示了他自己的程序，我觉得非常有意思。从那时起我就开始想编程了，先后借了很多相关的书。最初是学习编写HTML，之后又转向CGI。

《程序员》：你编写的第一个有意思的程序是什么，还有印象吗？

John Resig：高中时，有一阵我通过编写网站应用来挣钱。我还会做一些诸如网站设计的工作，当然没有专业人员做得那么好。记得曾编写了一个订早餐的网页应用，让那些熬夜无暇早起买早餐的人通过这个网页应用订餐。不过，这还算不上最有意思的程序。我的第一个最有趣的程序是在大学时编写的。那时我们要建立自己的时间表，以分配好上课时间，所以我编了一个选课的程序，它可以推荐出最优的选课结果，例如：如果你要选数学、计算机和除此之外的一门科学

课程，程序可以推荐出可在一天内完成三门课程的时间表，这样你就有六天的空闲时间去做自己想做的事情。

《程序员》：学生时代的生活和以后的编程生涯，两者之间有没有内在的联系？

John Resig：这说不太清楚，我喜欢编程，它可以让我完全理解一个东西。选择Web编程是因为自己愿意做别人不乐意做的事情。比如很多人不愿意处理同样的网页在不同浏览器下表现各异的问题。但这很有挑战，也很有乐趣。

《程序员》：能不能谈谈你在大学生活中印象最深的事情？

John Resig：对我来说最重要的事情，是那些计算机科学课程。大学期间我并没做很多Web编程，而是做了很多和数据库相关的工作。这是我感兴趣的方面。我还喜欢研究社区，做数据挖掘研究，其中就包括判断社交网站怎样增长之类，那时我还发表了两篇关于数据挖掘的论文。离开大学后，我又回到Web编程，尽管数据挖掘很有意思，但我还是觉得Web编程的吸引力更大。

《程序员》：大学最喜欢的计算机课程有哪些？

John Resig：我喜欢有挑战性的课。例如XML，我本来以为课程很简单，可以轻松得个A。结果老师第一堂课就把我知道的东西都讲完了，从第二节课开始所讲内容我完全听不懂，很难学，结果我得了B。但我还是很自豪，因为学了很多东西。

jQurey 背后的故事

《程序员》：向我们透露一些jQuery背后的故事吧。最开始您是怎样做的？

John Resig：做Web编程时，我非常讨厌浏览器的Bug，不同的浏览器有不同的Bug，而且数量非常多。于是我用JavaScript做了CSS选择引擎，之后还做了个动画引擎，都是自娱自乐。但与此同时我发现自己不能将制作的一些应用放到浏览器里。为了将应用放到Firefox浏览器中，我开始制作相关的API，以应用那个CSS选择引擎和动画引擎，这些最终成为了jQuery。几个月后，我将那些应用做进Firefox里，之后在IE里也可以运行。如今这仍是我的目标——让每个人都可以在网页里写点什么，并且写的东西能够在浏览器中顺利运行出来。

《程序员》：支撑jQuery的基本原理有哪些？

John Resig：原则很简单。作为一名程序员，我希望代码简洁，不希望在编程的时候不停地重复某些内容，设计jQuery的目标就是为了简化代码，使程序更高效。

《程序员》：jQuery是如何把简单和高效结合在一起的？

John Resig：很多人想直接做大项目，例如像Gmail、Yahoo!Mail之类。但实际上通过很简单的过程也可以解决大项目中一些很困难的关键点，化大为小，化繁为简，jQuery就是这样做的。

《程序员》：如果能重新再来一次，你会在哪些方面做出设计改变？

John Resig：我要改一些方法的名称。初期在命名上出现了一些失误，后来我花了很多时间才理解到jQuery应该是现在这样。有些事情如果一开始就做好，可能会少走很多弯路。

《程序员》：谈谈jQuery的研发过程吧。

John Resig：我们主要的资源是jQuery的Bug Tracker，有一个Team专门经营Bug Tracker。我负责修改这些Bug。最终我们有一个需要修改的Bug列表。然后有人改程序，提交给别人通过，并处理这些Bug。大多数时候都是我来改Bug，发布新版本。

《程序员》：中国的程序员该如何做才能参加到这个项目？

John Resig：直接加入就可以。所有人都可以看到内部的每次代码提交，然后更优秀的程

序员会加入Bug Tracker。他们能看到我们发现问题和处理问题的整个过程，从而学会怎样发布补丁。我们的小组一共20个人，只有3个代码的贡献者，其余17人都在做各种各样其他事情。jQuery网站是世界排名前700位的网站，所以有很多工作需要处理。

《程序员》：在中国有没有正式的文档网站？

John Resig：有jQuery.org.cn，但这不是正式的，不过里面有jQuery文档的翻译资料。我们急需有人帮助翻译这些文档。

《程序员》：jQuery的未来怎样？

John Resig：我们基本不加入新功能，目前大多数工作都是优化，让jQuery变得更快、更强、更容易理解。未来的工作也是优化，使jQuery功能更清晰化。

开源是JavaScript的出路

《程序员》：说说你学习JavaScript的过程？

John Resig：从高中时我就开始使用JavaScript，具体做什么记不大清了。大学时我加入一个研究工作组做一些商业项目，例如为一些地方公司设计网页。设计师的工作是用Photoshop处理网页图片，而我的工作是将这些图片用CSS展现在网站上，这需要在Firefox浏览器下呈现出相应的效果。这期间公司还让我做的事情是在网页上显示一个特殊的卷轴效果，所以必须用JavaScript编写。我一直在琢磨如何写这个脚本，直到有一天我改了脚本中某个元素属性，这个功能就实现了。我很高兴，开始系统学习JavaScript，之后就用它编程。

《程序员》：最开始你用的是什么库？

John Resig：用Prototype，这是2005年出现的一款非常棒的JavaScript基础类库，对JavaScript做了大量的扩展，而且很好地支持了Ruby on Rails。Prototype吸引我的一点是代码干净整洁。当我第一次看到Prototype的时候，很难想象JavaScript类库代码可以做到如此简洁。Prototype非常漂亮，让人产生用它写代码的欲望。

《程序员》：你认为什么时候是JavaScript库开源的最佳时机？

John Resig：JavaScript与其他浏览器语言有很多不同，它的特点是大家都可以看到源代码，所以我认为从一开始就要开源。如果你注意观察JavaScript的库，就会发现所有开源的库都挺好，而所有闭源的库都不行。其实现在基本已经没有人再用闭源的JavaScript库了。

《程序员》：我知道有些中国的公司开始设计新的框架和JavaScript库，并打算开源，你对他们有什么建议吗？

John Resig：开源是需要的，但最重要的是知道你的用户是谁，否则就会跟他们有距离。我设计jQuery时就遇到了类似的问题，要考虑究竟哪些人需要用我设计的类库。目前JavaScript还是有发展的空间，但是如果创造全新的东西就可能没太大必要。我对开源公司的建议是必须放弃自己的公司立场，把这个当成一个独立的项目，公司只是这个项目的一个用户。比如公司有一个需求，也要走正式的申请、Debug之类的流程。也许有些人会觉得这是公司的资源，但如果你要开源，就必须放弃控制。

《程序员》：对刚开始使用JavaScript的开发者有什么建议？

John Resig：刚开始使用JavaScript的开发者估计对浏览器方面的了解也很少。我的建议是用库。不要把时间花在那些浏览器的Bug上，一开始就直接用各种库。

《程序员》：你怎样看JavaScript的未来？谈一下服务器端的node.js？

John Resig：我对JavaScript的未来很乐观。因为网络和浏览器会存在很长时间，所以JavaScript和HTML一样会长久存在。20年内大家肯定会一直用JavaScript写Web应用。它是一种很特别的语言，在服务器端也可以用JavaScript。我

喜欢JavaScript的原因是它可直接用在浏览器上，它跟Python不一样，很少有语言可以直接用在浏览器上。在服务器端的node.js是很酷的东西，它让JavaScript脱离浏览器而存在。

优秀程序员的标准

《程序员》：谈谈你在Mozilla的工作？

John Resig：我以前是JavaScript程序员，2007年2月加入Mozilla，工作了几年。后来jQuery变得很大，我必须全力来做。所以向Mozilla的主管说我要做jQuery，他非常支持，认为这是很棒的事情。所以后来，我就可以将全部精力放在jQuery上。

《程序员》：你最常用的工具是什么？

John Resig：用浏览器，12~20种，不同的版本，不同的类型。我还要使用各类系统如Windows、Mac等。

此外Firebug是一款很棒的工具。

《程序员》：你现在是在本地还是在云端工作？

John Resig：我基本还是在本地，在云端工作的确很好，因为设定所有的模拟器是很困难的，所以我们都是设定一次，然后放到云端，这样能避免一些测试的问题。

《程序员》：你觉得好的程序员应该是怎样的？

John Resig：面试JavaScript程序员时，我一般问两类问题，一类是JavaScript的技术问题，另一类是浏览器Bug方面的。比如你最喜欢的浏览器Bug，要是他不能回答，就说明他做的工作不够。喜欢这个工作的人，都会有花两三天修复一个浏览器Bug的经历。

《程序员》：你觉得学习数学或者理论、算法是不是对程序员很重要？

John Resig：对有些程序员是，但对JavaScript程序员则不一定，这取决于你要做什么工作。在大学时我做数据挖掘方面的工作，需要很强的理论背景。在用JavaScript时，我学了一些语言基础方面的东西。当然多学知识肯定会让你有更好的理解。算法有时不一定都能用上，但熟悉语言的基础理论会对自己有很大帮助。

HTML的未来

《程序员》：HTML、JavaScript在将来是不是会占统治地位？

John Resig：现在已经处于统治地位了，以后浏览器和web只会更流行。它是网络的核心，不像Flash不能运行在iPhone上。

《程序员》：HTML5会成功替代Flash吗？

John Resig：大家要用Flash主要是为了视频、游戏。但现在主要的视频网站都已经转入HTML5，我认为随着更多的浏览器支持HTML5功能，Flash就会更加无关紧要。

《程序员》：越来越多人在用HTML和JavaScript，会不会取代所有的本地应用？

John Resig：是的，因为用它们可以开发很多功能。HTML会不会取代所有的本地应用，关键还是看要用什么功能，比如你不能用HTML编写浏览器，还是要使用底层的语言。当然不排除但首先Web技术可能变得更厉害，厉害到能开发浏览器的程度，就可以编浏览器。要让所有浏览器都能支持，这是未来的关键。

记者后记：

在专访后不久的2011年5月，John Resig离职Mozilla，加入在线教育集团Khan Academy。在那里，他仍继续从事jQuery相关工作，同时负责该组织的开源项目及未来的iPad应用的开发。Khan Academy是一个非盈利组织，正在尝试对“学生的学习方法及老师的教育方法”进行一次彻底变革。John Resig的加入，意在为教育提供更好的工具，让更多的人参与到开源中来，为提升教育质量贡献一份力。

行业律师的IT编程之路

专访北京市盛峰律师事务所主任律师于国富

记者 / 陈秋歌

成功需要天时、地利、人和。于国富认为，对于个人，要做到“以己之长，克人之短”，对所追求的目标要“锲而不舍”。

于国富是北京市盛峰律师事务所的主任律师，同时也是一位计算机编程爱好者。他将自己的专业与个人爱好很完美地结合在一起，开创了一条适合自己的成功之路。

“老老实实”过童年

于国富，河北人。用他的话来说，他有一个“老老实实”的童年。比别人早上学的他，爱学习，在父母及邻居眼里是一个听话、正直的孩子。从小受父亲影响，对赌博深恶痛绝，不抽烟、不喝酒。无不良因素的干扰，他把更多的时间放在学习上，所结交的也均是志同道合、无不良嗜好的朋友。

他视学习为一种极具乐趣的事，父母也从没给他太多的压力。因为平和的心态及优良的学习成绩，他如愿考上了西南政法大学法律系。虽是一个文科生，但于国富中学时的数理化成绩也十分优秀。在高二分科前，他的数学还获得过全校前三名的好成绩。他像其他男孩子一样，视拆卸、组装一些小的家电机械为童年的一大乐事。

学习机开启编程新乐趣

在电脑尚不普及的20世纪90年代初，小霸王学习机是孩子们的理想玩具。1992年高考结束后，由于高考成绩不错，“作为奖励，父母花了486元为我买了一部小霸王486学习机。当

于国富坚信“以己之长，克人之短”

时对于一个普通的农民家庭来说，486元是蛮大的一笔数字呢。”回想起来，父母的关怀和重视，至今仍让于国富感激不已。

利用这部学习机，于国富学习了当时十分流行的五笔输入法，并初步接触编程。当时的学习机里内置有Basic语言，利用它，可以把自己想象中的事物通过学习机编写出来，对于他来说这确实是件十分神奇的事，令他有一种无可比拟的成就感。从此他对编程产生了浓厚的兴趣。

到了大学，他才接触到真正的计算机，学习到WPS、CCED等软件，同时对相关软件编

程知识也有了更深入的了解。

毕业从教，深入了解网络技术

大学毕业后，于国富被分配到一所理工院校当老师。这是一个“卧虎藏龙”的地方，他广交朋友，其中不乏优秀的计算机专业青年老师。出于对计算机的喜爱，于国富经常利用业余时间，向他们讨教计算机、编程方面的知识。他还曾与几位青年老师在学校里最早搞起了网络教育：一方面他把自己的教案提前放在服务器上，供上课时直接下载用于授课；另一方面他还搭建了自己的教学网站，学生们利用上机时间，浏览上节课内容，以达到动态教学的目的。

当时，能拥有一台电脑是件比较奢侈的事情。很幸运，于国富所在教研室配有一台“先进”的586电脑。对此他爱不释手，用他的话来说，“这台电脑一直被我霸占着。”碰到其他教师录入教案，他都主动要求帮他们录入，以便获得更多上机时间。

在这段从教经历中，于国富深入了解了网络技术，包括集线器、交换机、服务器等重要网络硬件；包括架设网线、实现网络共享、权限设置以及网页制作等技术。同时他还初步学习了C程序设计语言，“利用自己老师的身份去做其他老师的学生”。

入行律师，体验IT编程之路

2000年，于国富离开了这所理工院校。不久便进入了盛峰律师事务所，开始专注于网络知识产权这一服务领域。

网络知识产权类案件的专业性非常强，要求律师有一定的IT基础，其工作流程也相当严格。每个案件所提交的源代码及说明文档都要整理成固定的、符合条件的格式，对于其他律师来说，此类案件可能很棘手，但对于国富来说，就是一份“美差”。因为这是他的兴趣所在。

除了处理相关案件以外，于国富还十分关注盛峰律师事务所的流程化管理，他最先倡导并参与开发了适应该事务所的OA系统。

起初，于国富请来几位从事技术开发工作的朋友，用了半个月时间搭建好基本架构。但该系统却无法适应律师的工作流程，比较难用。几位朋友毕竟非律师出身，对其中诸多细节不太了解。于是于国富凭借自己对律师行业的熟知及对IT专业知识的掌握，开始了大框架下小模块的具体开发。

律师工作的最大特点就是提成制。一个案件从立案、批准、进行中、完成到核准的整个业务工作流中，都伴随着律师报酬的产生。对此于国富分别开发了案件子系统和薪酬子系统。案件一经设立，案件子系统就会产生相应案件的业务工作流，同时薪酬子系统便会伴随案件业务流的不同阶段产生相应的报酬。这两套子系统，可以使律师对所拿薪酬清晰明了，避免了众多因薪酬不明而带来的诸多误解，使律师事务所真正做到了人和。

于国富认为案件和薪酬这两个子系统是该OA系统的核心所在，在此基础上，他又延伸开发了请假、文案管理、办案费用等子模块。在他眼中，“这是一套用起来十分顺手、适用于律师事务所的流程管理系统。”同时，于国富也深刻体会到，系统成功的最关键点在于该系统要服务于其服务对象，而不是让服务对象服务于它。现在于国富正在考虑把当前基于ASP的OA系统迁移到PHP架构上，让其更高效。

结语

法律专业出身的于国富，利用他在IT技术方面的优势，开辟了个人事业发展的新途径，并正一步步走向成功。回顾走过的历程，于国富总结说：经验不外乎两点，一是“以己之长，克人之短”，在规划自己未来事业发展方向时，他并没有遵循别人固有的标准，而是结合自己的优势将其充分发挥出来。方向选对了，只要加以努力，就会逐渐走向成功。另一点则是“锲而不舍”。虽然有时编程十分枯燥，尤其是遇到编码测试通不过时，难免让人心烦气躁，但于国富并没有因此放弃，因为他相信，行百里者，半九十。

Mac OS X 背后的故事（四）

政客的跨界

文 / 王越

《Mac OS X背后的故事》系列文章将为大家介绍Mac OS X的发行版本、技术历史、相关人物等内容。本文是系列连载的第四篇。

2000年，美国总统大选，由于选票设计问题，时任美国副总统的 Al Gore 败北。2000年12月13日，在一番重新计票的大折腾不起作用后，曾经意气风发的 Al Gore 拖着疲惫的身子，走上讲台，发表了认输讲话（Al Gore《2000 Presidential Concession Speech》），从此退出政坛。一般国家领导人的退政生活其往往松愉快，出出日记，学用哲学，或者像多才多艺的李岚清不但去各地推广古典音乐，更是玩起了篆刻（《南方周末》2006年05月11日《老常委的卸任生活》），克林顿先生都成立个基金会来帮助社会预防和治疗爱滋（http://en.wikipedia.org/wiki/Bill_Clinton）。 Al Gore也没闲着，他找到了让他感兴趣的去处——Apple总部，并成为董事之一。

Mac OS X和Al Gore的双赢

2003年5月19日，Apple的启示中罕见性地登出了《前总统Al Gore加入 Apple 董事会》的快讯（http://www.apple.com/pr/library/2003/mar/19gore.html）。文中提到，Al Gore总统是一个正宗果粉，他一直用Mac计算机，而且还会用Final Cut Pro来编辑他的视频。Al Gore也不掩饰他对Apple技术的热爱，他表示对Mac OS X的开发极感兴趣，并且也对Apple在开放源代码运动中的贡献喜闻乐见。他虚心地说，想在这个让Apple起死回生的董事会好好观摩并学习。

前总统Al Gore的加盟使苹果一跃成为电子环保产品的代言人

苹果公司的CEO Steve Jobs表示Al Gore曾经管理过世界最大的组织——美国政府，期间显示出的经验和智慧对苹果公司是笔巨大的财富。Al Gore将成为出色的董事会主席，苹果将以他把苹果公司作为职业生涯的开始为荣。

这之后，Al Gore在Apple内部的决策究竟起了什么作用，和Mac OS X的开发有何关联，在正式的渠道很少有史料，但是他后来的各种公开活动，却给Mac OS X的技术做足了广告， 而且很多证据表明，他正是使Apple从被绿色人士攻击的众矢之的的状态，成为业界注重电子产品环

保领头羊的主要推手。

Al Gore重新进入普通人的视野是在2006年，他推出了自己参与制作和演出的纪录片《An Inconvenient Truth》（《难以忽视的真相》）和同名书籍。这部长达94分钟的影片，在西方国家引起了广大的回响，以Al Gore的一场演讲和人生的回忆作为两条主线，详细、科普地向民众介绍了全球变暖问题的科学证据及美国政府掩盖问题的真相。该片以发人深省的立意、详尽的科学数据、平实的讲演风格，加上苹果高超的技术，而获得了广泛的好评并一举获得年度奥斯卡最佳纪录片奖，使得这位美国前副总统摇身一变，成为好莱坞明星。

为什么单单一场简单的讲话，就能做出一部电影，还能得到奥斯卡这样学院艺术奖的亲睐？是因为讲话内容无懈可击么？事后有很多科学家站出来表示，虽然影片内容有积极的意义，但其实也有很多被夸大的科学数据、假设和结论（http://www.johnstonsarchive.net/environment/gore.html）。试思索，该片之所以成功，甚至成为诸多演讲培训机构的重要分析案例（http://www.presentationzen.com/presentationzen/2006/05/al_gore_another.html），除了数据、观点、论述外，还有以下几个原因。

首先，这场演讲由苹果主导的技术和艺术的设计。Al Gore向来以说不清想表达的内容而著称。他经常因为讲得过于专业或者缺乏好的表述方法以致于民众完全不懂他在讲些什么。他的早期讲话用现在的眼光看就是个少将体，比如“互联网...网...我...这个...那个...那个...怎么说呢...我想这个...这...这...这...我啊...我啊...就是说...互联网是我发明的！”（http://en.wikipedia.org/wiki/Al_Gore_and_information_technology，另见Authors@Google: Garr Reynolds, http://www.youtube.com/watch?v=DZ2vtQCESpk）因此作为苹果展现公司软实力的重要机会，苹果非常重视这场讲话，请公司的图形设计小组带领完成各种所需设计，苹果甚至特地请来了专业的设计公司Duarte来进行讲稿和讲话内容的安排。因此，不管是内容安排、图形设计还是技术支持，Al Gore都有强有力的后盾，他们能够帮助Al Gore完成任何想达到的目标。不论是FinalCut还是Keynote，一旦缺少任何Al Gore想要的功能，Apple都可以给他开小灶实现。在片末的走马灯字幕中，有大量Apple的Keynote组、Final Cut组和图形设计组的员工名字，以示鸣谢。

其次，上面这些资源的相互合作，也使得Al Gore的这场讲话的讲稿被精心制作，体现了精心设计的电子稿演讲所能达到的最高成就。苹果公司向来重视演讲，也是各大企业中最会通过演讲来营销产品的公司。每年的MacWorld和WWDC的 Steve Jobs 讲话都会吸引百万人在计算机前观看。每场讲话都好戏连连，台下的观众的欢呼和掌声不亚于著名歌星的演唱会。这种风格显然给Al Gore的讲话风格带来很大的影响。在影片中，观众看不到一个传统的bulletpoint（PowerPoint用户常爱使用的表示讲话结构的方法），取而代之的是高清的照片、视频，来展现环境的严峻性。观众不再会为枯燥无味的技术词语而搞得昏昏欲睡，因为屏幕上的一切都是如此真实，各种科学现象由动画效果配合，使其浅显易懂。另外，所有的数据、图表都精心使用软件制作，使其一目了然，表现准确而美观大方，而且Al Gore时而还会玩些小噱头，比如讲到现在的温室气体浓度是多么高时，他甚至爬上工作人员为他准备的升降机，升到舞台顶端，来告诉观众，数据已经突破图表的顶端了。现在距笔者观赏完这部影片，已经五年过去了，但影片中的灾难场景、冰川融化的影片段落、海平面上升的计算机模拟、二氧化碳浓度的数据图表，至今都记得一清二楚，足以见得其表现力是何等深入人心。甚至有人在调侃他在2000年的竞选演说是怎么回事？难道就是缺少了这些科技元素？

最后，Mac OS X的各项技术也是这部片子的重要保证。Duarte 公司的 Ted Boda表示（该幻灯片的设计师之一），Mac OS X系统本身的反锯齿功能把文字、图片、矢量图标表现得栩栩如生，使得幻灯片充满美感。QuickTime技术作为Mac OS X的一块重要基石，又使得Keynote不需任何插件就能引入任何图片和影像，所以类似使用Illustrator、Photoshop、AfterEffects等软件做出的图片、影像或动昼，不需要任何转换过程就能直接拖到Keynote中。哪怕1920x1080的高清视频，都可以轻松插入，流畅播放。他们组根本想象不

出在Windows上使用PowerPoint会成什么样子。

可以说，没有Mac OS X，就没有这部电影。

而实际上这部电影的作用远胜过任何一部Apple公司的广告。片中Al Gore时时拿着PowerBook的笔记本，在办公室用Safari查网页，字体渲染真实而美观，甚至在车上都不忘打开笔记本用Keynote做几张幻灯片，就更不用说电影中Keynote幻灯片曾经迷倒多少Windows用户了。向笔者推荐这部电影的好朋友了解到这些全是Apple技术的功劳时，拥有一台Mac就成为其人生梦想。

环保卫士的Apple之路

作为环保人士，Al Gore对Apple的策略的影响也不容忽视。Apple向来被各环保组织长期批评，即使Apple长年不断地改进这方面问题，但绿色人士依然不买帐。哪怕在稍后的2007年，也仍有包括GreenPeace在内的七十多个组织联名写信给Al Gore，敦促Apple更重视环境问题，信中指责Apple仍在大量使用PVC和BFRs等对环境有害的材料，也不注重对自家产品的回收。由于Al Gore是Apple董事会成员，使得这个问题受到了Apple的广泛关注。Apple在07年后史无前例地迈开大步，大力推广环保计划（要求全世界的IT制造商们逐步弃用PVC等有毒的化学用品进行生产），让Apple一跃成为注重电子产品环境保护问题的领头羊。

从制造材料上，2007年8月发布的iMac成为分水岭。这款产品的设计主要使用可完全被回收的玻璃屏和铝外框，减小了塑料等不环保物质的使用，此后苹果一发不可收拾，把这项革命进行到底，从手机到笔记本，都全番设计。2008年的MacBook Air引出的Unibody技术是这场革命的代表产品，不但在外观上还是工程上做到极致，在环保上更是让各绿色组织无可挑剔。

在造势上，Apple现在每项主要产品的都有“环境”的标签页，从制造、运输、耗电、回收等性能情况分产品详细列出。Apple甚至在包装上都动足脑筋，尽量减少每个产品的包装，使得同一架飞机可以运输更多的产品，从而在运输相同数量产品的情况下减少飞机温室气体的总排放量。

Mac OS X的各项节电功能的开发更是不用说了。休眠、调整空闲时的屏幕亮度、硬盘转速等常规功能自然越做越好。而系统的多项技术能使程序更优地分配使用中央处理器和显示卡。甚至系统还能在用户打字时，每两键之间的空隙减少处理器的占用从而节省击键之间的功耗，这使得 Mac OS X不但更节约能源，笔记本的电池使用时间也不断提高。而这一切的变化，和Al Gore似乎都有着千丝万缕的联系。

由于《An Inconvenient Truth》中的讲话让Al Gore的观点深入人心，同时也对美国政府在京都议定的决策产生重大的压力，挪威诺贝尔委员会决定把2007年的诺贝尔和平奖颁给了Al Gore，以表彰其在全球环境问题方面的努力，同时苹果的主页上全版刊发新闻，以示祝贺。贺词如下：

Al has put his heart and soul, and much of his life during the past several years, into alerting and educating us all on the climate crisis. We are bursting with pride for Al and this historic recognition of his global contributions. (Al Gore在过去几年殚心积虑，全身心地投入对公众关于气候危机的警示和教育中。我们为他这次所得的荣誉和他全球性贡献的历史性承认感到无比自豪。)

或许，由于Al Gore在计算机领域的一贯低调（他也是Google的高级顾问），他在这些企业的工作很少被报道出来，但是他在政界的跨界身份是显而易见的。Al Gore在他的人生道路将何去何从，我们不得而知，但是从各种媒体信息的披露可以看出，Al Gore对计算机事业的热衷，对环保问题的投入，可能是美国历任领导人中最突出的。

王越

清华大学建筑学院毕业，现居美国。中国著名TeX开发者，非著名OpenFOAM开发者。

责任编辑：白羽中（baiyz@csdn.net）

分布式计算领域的哥德尔

Eric Brewer

文 / 苏椰

2010年3月，ACM宣布将2009年度“ACM-Infosys”奖颁发给加州大学伯克利分校的Eric Brewer教授，以表彰他在可扩展互联网等领域所做的贡献。Brewer在伯克利获得了计算机科学学士学位，随后在麻省理工学院获得计算机科学硕士和博士学位。32岁时，他拿到了伯克利的终身教授职位，这个职位是很多人的毕生梦想，而他却初出江湖就将其纳入囊中。这到底是个什么人？他到底做了什么事?

要说Brewer教授的工作，我们要从80年前的另一个人说起。1930年，著名数学家希尔伯特发表了他著名的退休演讲，其中有一句话鼓舞了一代数学家：“我们必须知道，我们必将知道。”这句话所指的，是他的“希尔伯特计划”，也就是证明数学体系的完备性、相容性和可判定性。数学家们为这美好的未来而不懈奋斗，然而这场美梦却被一个人打破了。1931年，哥德尔证明了，任何完备的系统都是不相容的，而任何相容的系统都是不完备的，二者不可得兼，这就是哥德尔定理。这个证明彻底摧毁了希尔伯特计划，使数学变成了一个残缺的世界。说到这里，不妨说一点题外话，希尔伯特提出的三个问题，哥德尔解决了前两个。那么剩下一个“可判定性”呢？这个问题是由艾伦·图灵在1936年的论文中首次解决的，也正是在这篇论文中，他提出了图灵机，后来成为现代计算机的理论模型。

读者也许要说，这80年前的事情，跟年轻的Brewer有什么关系啊？别急，马上我们就将看到，Brewer有着一个几乎同构的故事，他俨然就是分布式计算领域的哥德尔。对于一个分布式计算服务来说，有个CAP原理，包含三个最重要的属性：相容性（Consistency），读操作是否总能读到前一个写操作的结果，即是说在分布式环境中，多点读出的数据是否相容。可用性（Availability），访问数据的性能。分区容忍性（Partition tolerance ），数据的分区特性，对系统性能的影响程度。

对于一个分布式计算系统来说，这三个属性非常重要。因此，计算机科学家们一直在拼尽全力地寻找一种方法，能够实现同时满足CAP三者的完美分布式系统。但是就在大家都忙得热火朝天的时候，2007年，如同当年的哥德尔一样，Brewer教授站出来指出：CAP永远不可能同时满足，提高其中任意两者的同时，必然要牺牲第三者。这就是Brewer的CAP定理，后来由麻省理工学院的两位科学家证明。这个定理告诉大家，不要再浪费时间去研究如何兼顾了，因为这根本就是不可能的，只能根据具体应用，来决定如何在三者之间进行取舍。CAP理论为很多巨型的数据中心，尤其是现在很多的SNS网站提供了有力的理论指导，比如说Google的BigTable系统就是一个牺牲了A的典型例子。有意思的是，Brewe今年5月在Twitter

上透露，他利用学术休假的时间已经开始在Google公司领导下一代基础设施的设计了。就在提出CAP理论的同一年，Brewer当选美国国家工程院院士，并于次年当选ACM Fellow。

除CAP定理外，Brewer还有很多其他成就。1995年，他与人共同创建了著名搜索引擎公司Inktomi，曾是纳斯达克100指数成分股，后来被Yahoo!收购。2000年，他协助美国联邦政府做网站，建立了usa.gov并上线。他还非常关心发展中国家和贫困国家的信息化进程，提出了WiLDNet等很多概念，旨在通过技术方法，帮助人们以更低的成本获得无线通信能力。他的系统被部署用于印度的远程医疗，有超过20000名眼科患者通过该系统重新获得光明。

作者:
西乔
设计师，项目经理。06年起携创业团队从事Web技术外包开发及产品咨询顾问。
如果你有什么好玩的关于程序员的故事、对话、代码，愿意通过漫画的形式分享，
请给西乔发邮件：arthur369@gmail.com